

**机器学习**

**贷款违约预测报告**

|  |  |
| --- | --- |
| 学院： | 计算机科学与技术学院 |
| 专业： | 人工智能 |
| 班级： | 19级智能班 |
| 姓名： | 陈怡然 |

目 录

[一、写在前面 1](#_Toc12974769)

[二、赛题介绍 2](#_Toc12974772)

[1. 赛题背景 2](#_Toc12974773)

[2. 任务 2](#_Toc12974774)

3. 赛题数据 …………………………………………………………………………………………………………………………...........2

4. 评测标准……………………………………………………………………………………………………………………………………..4

[三、问题分析 5](#_Toc12974780)

[1. 类型 5](#_Toc12974781)

[2. AUC 5](#_Toc12974782)

[3. 数据 5](#_Toc12974783)

[四、赛题整体流程 7](#_Toc12974784)

[五、数据EDA 8](#_Toc12974789)

[1. 总体分布 10](#_Toc12974790)

[2. 数据类型分析 13](#_Toc12974791)

[六、特征工程………………………………………………………………………………………………27](#_Toc12974805)

1. 重复值处理………………………………………………………………………………………………………….………………..…..27

2. 缺失值处理………………………………………………………………………………………………………………………………..28

3. 异常值处理………………………………………………………………………………………………………………………………..29

4. 时间数据处理 ……………………………………………………………………………………………………………………..…..30

5. 特征交叉 ………………………………………………………………………………………………………………………………….31

6. 特征编码 ………………………………………………………………………………………………………………………………….32

7. 特征选择 ………………………………………………………………………………………………………………………………….35

8. 对训练集处理 ………………………………………………………………………………………………………………………….37

8. 对测试集处理 ………………………………………………………………………………………………………………………….39

[七、建模分析 40](#_Toc12974811)

1. 模型考虑………………………………………………………………………………………………………….…………………...…..40

2. 尝试过程………………………………………………………………………………………………………….……….…………..…..43

八、结果 51

九、总结与收获 ……………………………………………………………………………………………………………………………………53

十、代码 ………………………………………………………………………………………………………………………………………….……54

1. **写在前面**

本次机器学习大作业我选择了天池新人赛项目贷款违约预测项目。初看到项目介绍感觉和华为实验指导书上的很多东西是类似的，就是一个数据量大一些的二分类罢了。但是毕竟刚开始接触机器学习，对于一些算法的理解和使用还是不够熟悉，或者说是还没有很好地掌握机器学习，所以在一开始还是很蒙的状态。

不知道如何下手，在查询了很多资料后，渐渐地步入正轨。

从看到题目到完成大约用了两个星期的时间，中间走了不少弯路，一开始做的数据处理，特征工程等数据清洗地过于干净，导致最后的模型无论怎么选择分数都不是很高。后来发现项目给出的数据就已经很干净了，是很符合实际贷款情况的数据，所以不用清洗太多地数据，即使有一些是在学习过程中通常会处理的情况。最后简化了数据处理和特征选择后，分数提升地就比较大了。也学习到了理论要服务与实际的道理。

总的来说大作业很难，我做的很艰辛，收获很丰富，不想挂科，不想再来一遍了，求求了！！！

**二、赛题介绍**

**1.赛事背景**

赛题以金融风控中的个人信贷为背景，要求选手根据贷款申请人的数据信息预测其是否有违约的可能，以此判断是否通过此项贷款，这是一个典型的分类问题。

**2.任务**

预测用户贷款是否违约。

**3.赛题数据**

赛题以预测用户贷款是否违约为任务，数据集报名后可见并可下载，该数据来自某信贷平台的贷款记录，总数据量超过120w，包含47列变量信息，其中15列为匿名变量。为了保证比赛的公平性，将会从中抽取80万条作为训练集，20万条作为测试集A，20万条作为测试集B，同时会对employmentTitle、purpose、postCode和title等信息进行脱敏。

* 字段表





**4.评测标准**

提交结果为每个测试样本是1的概率，也就是y为1的概率。评价方法为AUC评估模型效果（越大越好）。

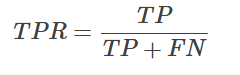
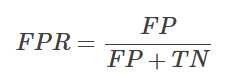
**三、问题分析**

**1、类型**

通过对赛事背景的理解，可以很清楚地得知此为一个二分类问题，对于每个样本都需要得到其y为1（即贷款违约）的概率。

**2、AUC**

其评测标准为AUC。AUC在机器学习领域中是一种模型评估指标。根据维基百科的定义，AUC(area under the curve)是ROC曲线下的面积。在二分类（0，1）的模型中，一般我们最后的输出是一个概率值，表示结果是1的概率。需要一个阈值，超过这个阈值则归类为1，低于这个阈值就归类为0。所以，不同的阈值会导致分类的结果不同，也就是混淆矩阵不一样了，FPR和TPR也就不一样了。所以当阈值从0开始慢慢移动到1的过程，就会形成很多对(FPR, TPR)的值，将它们画在坐标系上，就是所谓的ROC曲线了。

****

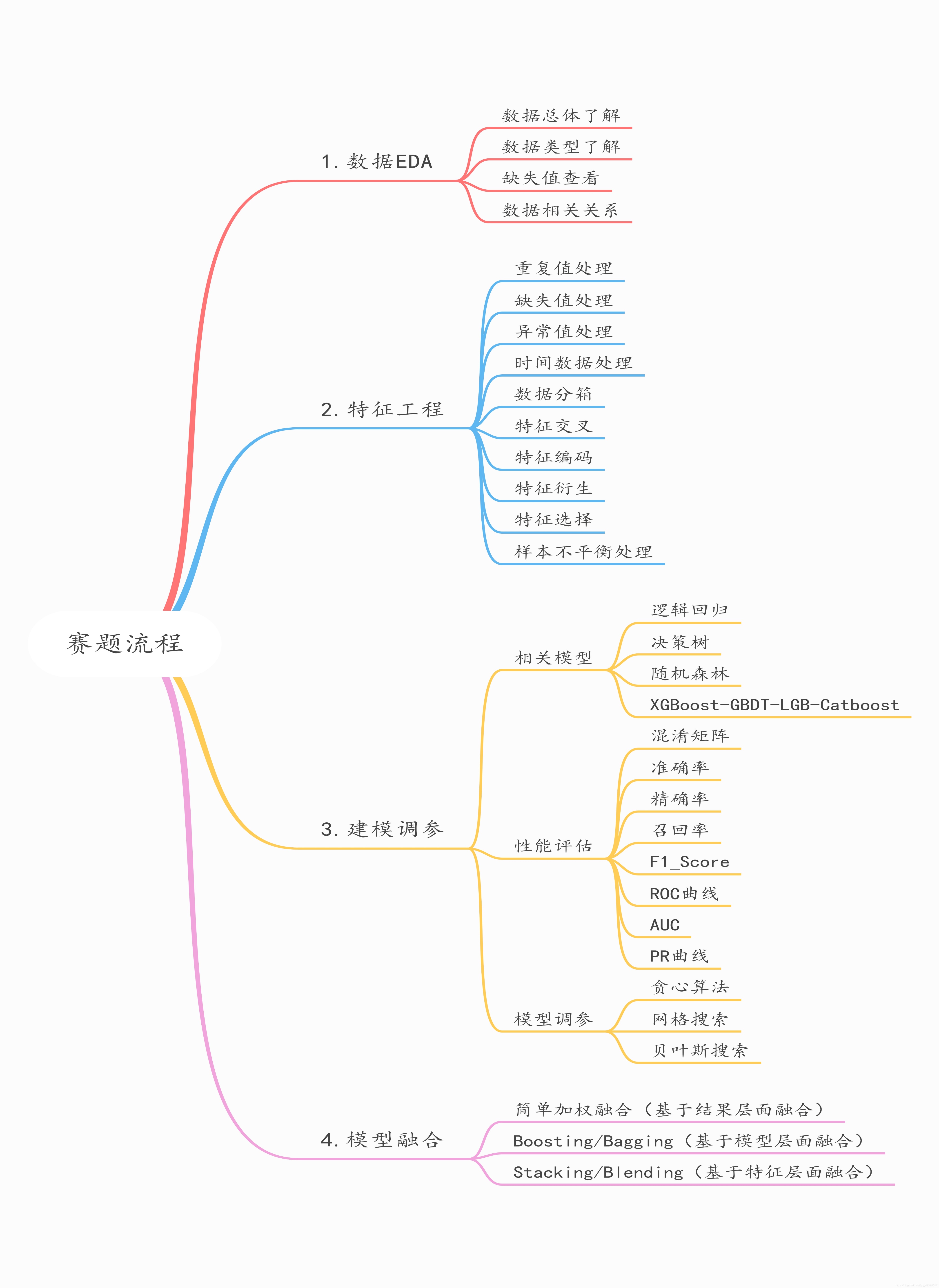
****

**3、数据**

数据都是来自信贷平台的贷款记录，所以数据中的很多特征都是在实际中很重要或者是对于结果预测有很大影响的特征，所以在进行特征选择时需要注意不要将数据处理得太过于干净，否则将会影响分数，就像我的分数一样，在一开始特征选择和数据处理地过于干净，反而导致了最后分数较低，且不断地更换模型也无法使分数提升地很快，所以在重新简化特征选择和数据处理后，分数提升地比较快。



**四、赛题整体流程**

****

**五、数据EDA**

数据探索性分析是对数据进行初步分析，了解数据特征，观察数据类型，分析数据分布等等，为后续特征工程，以及建模分析都特别重要。

分析数据中每个字段的含义、分布、缺失情况；字段表示什么含义、字段的类型是什么、字段的取值空间是什么、字段每个取值表示什么意义；字段整体的分布，分析字段在训练集/测试集中的分布情况；字段缺失值的分布比例，字段在训练集/测试集的缺失情况；分析数据中每个字段的与赛题标签的关系；分析数据字段两两之间，或者主者之间的关系。



首先导入模块和数据集：

import warnings

warnings.filterwarnings('ignore')

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

import datetime

import statsmodels.formula.api as smf

from sklearn.preprocessing import LabelEncoder

from sklearn.model\_selection import train\_test\_split

from sklearn.feature\_selection import chi2

from sklearn.preprocessing import MinMaxScaler

from lightgbm import LGBMClassifier

from sklearn.feature\_selection import SelectKBest

from sklearn.model\_selection import StratifiedKFold,KFold

from sklearn.model\_selection import cross\_val\_score

import xgboost as xgb

import lightgbm as lgb

from catboost import CatBoostRegressor

from sklearn.ensemble import GradientBoostingClassifier

from lightgbm import LGBMRegressor

from catboost import CatBoostRegressor

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import roc\_auc\_score,accuracy\_score,f1\_score,log\_loss

plt.rcParams["font.sans-serif"]=["SimHei"]#正常显示中文

plt.rcParams["axes.unicode\_minus"]=False#正常显示负号

from jupyterthemes import jtplot

jtplot.style(theme='onedork') #选择一个绘图主题

train=pd.read\_csv('G://classStudy/II 2/机器学习/大作业/贷款违约预测/train.csv')

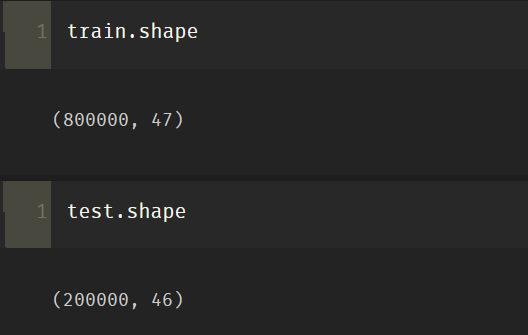
test=pd.read\_csv('G://classStudy/II 2/机器学习/大作业/贷款违约预测/testA.csv')

查看部分数据：

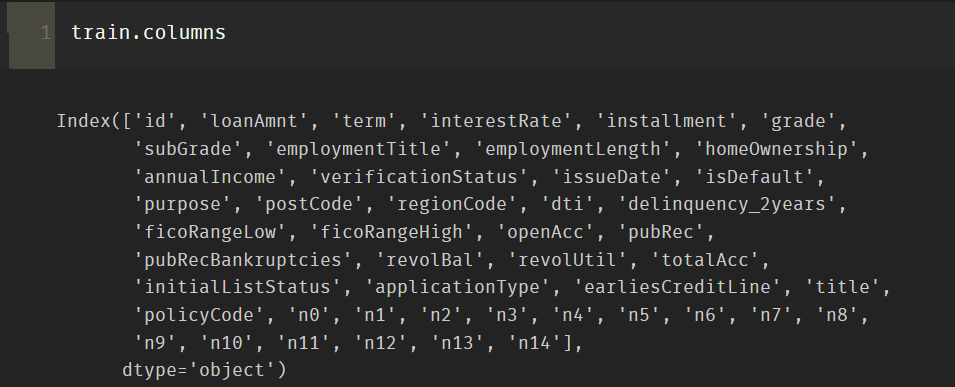


1. **总体分布**

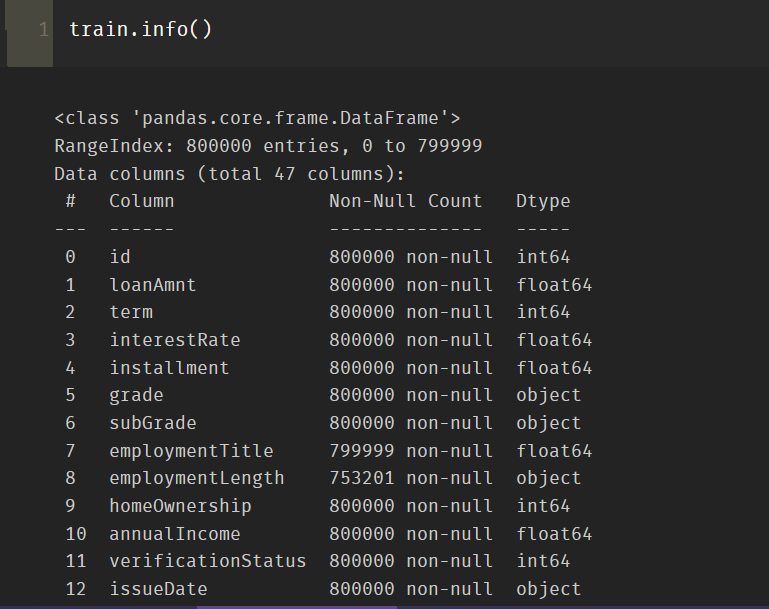
前面提到，整个数据包括80万条训练集，20万条测试集A，20万条测试集B。另外训练集中有47列，其中包括46个特征列，1个标签列测试集中只有46个特征列。

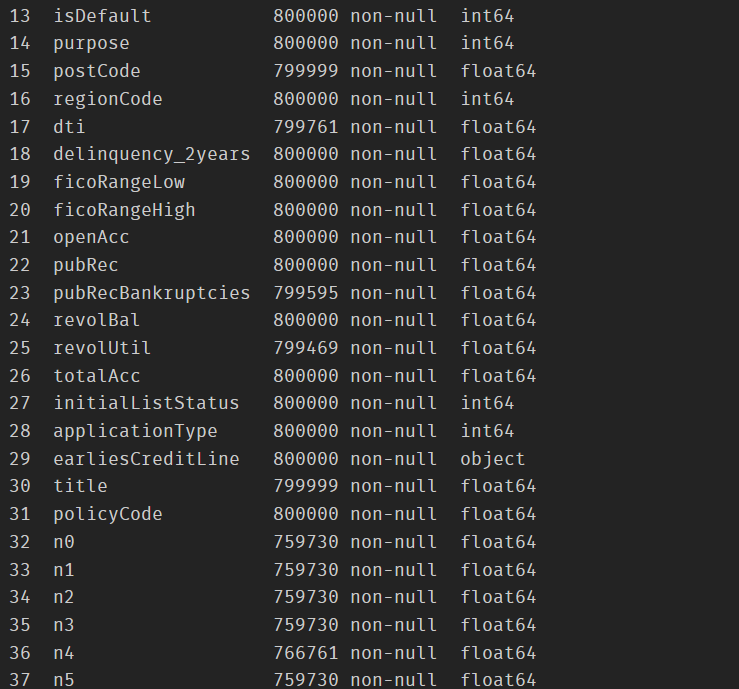


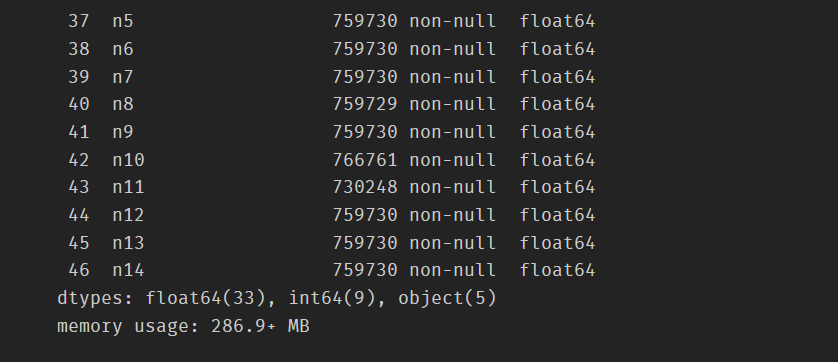
查看特征名：



接下来查看数据集的一些基本信息（缺失情况、类型…）：

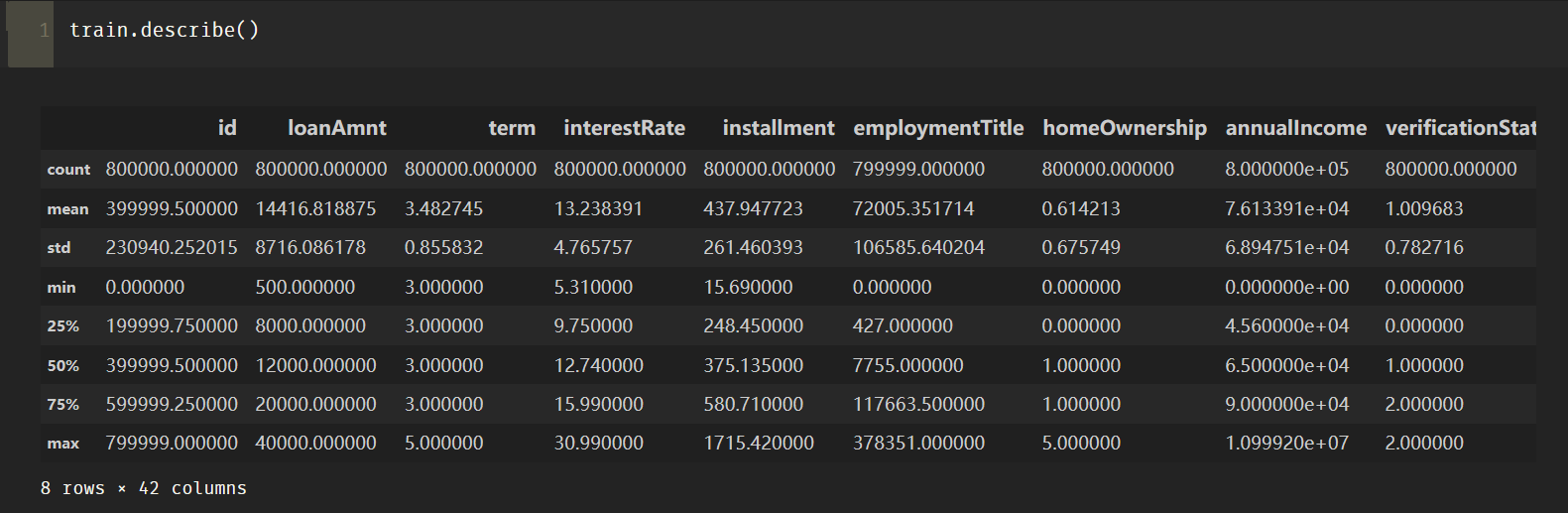






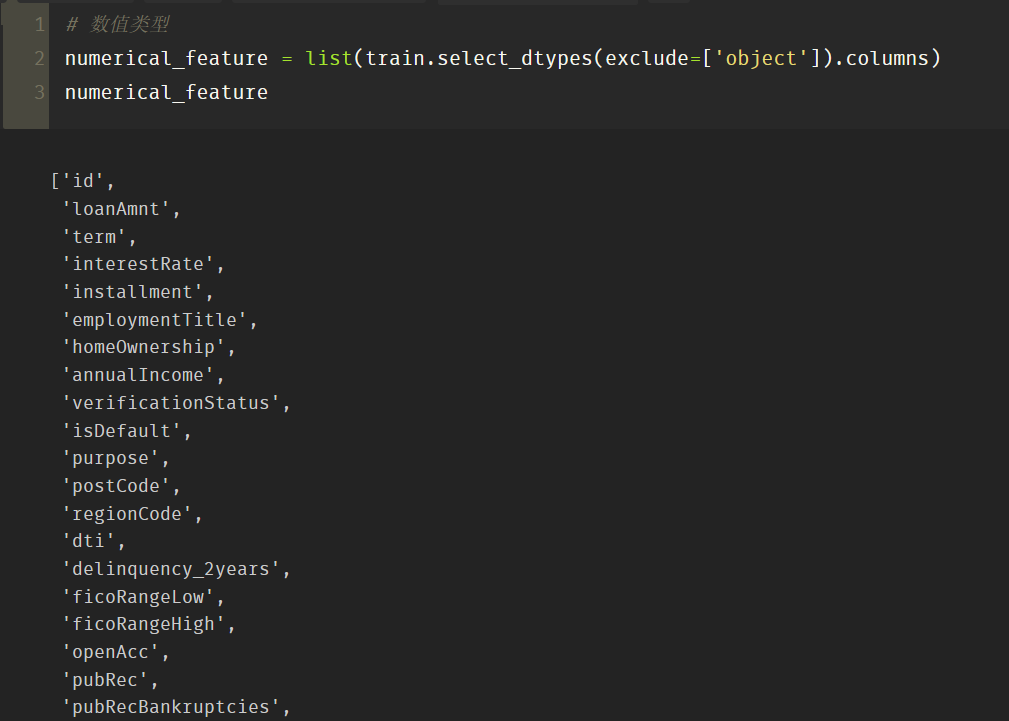
可以看到，许多特征存在缺失，特征的类型有dtypes: float64(33), int64(9), object(5)。

接下来查看一下数据的描述性分析：



1. **数据类型分析**

2.1、数值类型（连续变量、离散型变量和单值变量）：





由于数值类型又可以分为连续变量、离散型变量和单值变量因此接下来把数值型中的连续型变量和离散型变量区分开来：

# 连续型变量

serial\_feature = []

# 离散型变量

discrete\_feature = []

# 单值变量

unique\_feature = []

for fea in numerical\_feature:

temp = train[fea].nunique()# 返回的是唯一值的个数

if temp == 1:

unique\_feature.append(fea)

# 自定义变量的值的取值个数小于10就为离散型变量

elif temp <= 10:

discrete\_feature.append(fea)

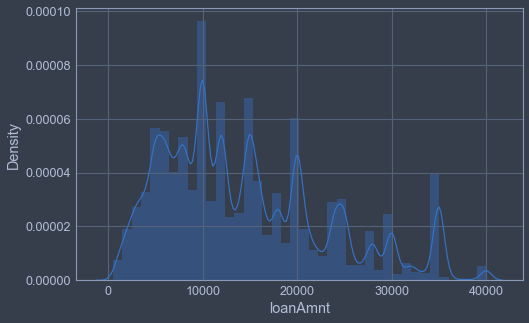
else:

serial\_feature.append(fea)

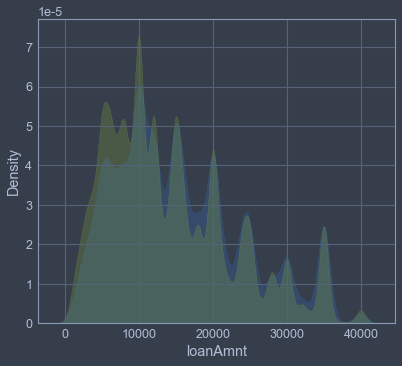
* 对于连续型变量

查看某一个数值型变量的分布，查看变量是否符合正态分布，如果不符合正太分布的变量可以log化后再观察下是否符合正态分布。正态化的原因：一些情况下正态非正态可以让模型更快的收敛，一些模型要求数据正态（eg. GMM、KNN）,保证数据不要过偏态即可，过于偏态可能会影响模型预测结果。

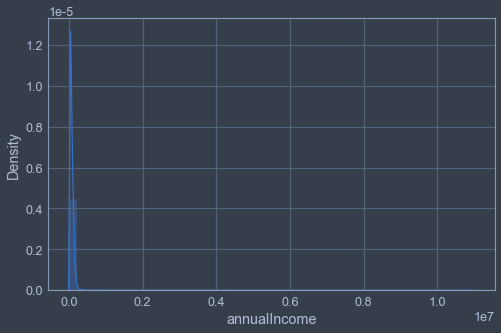
可以单独查看一下贷款金额 loanAmnt 的分布情况：



对于违约与不违约两类样本的贷款金额分布情况：



单独查看一下年收入的分布情况：



* 对于离散数据

离散型特征可视化呈现：

df\_ = train[discrete\_feature]

sns.set\_style("whitegrid") # 使用whitegrid主题

fig,axes=plt.subplots(nrows=4,ncols=2,figsize=(8,10))

for i, item in enumerate(df\_):

plt.subplot(4,2,(i+1))

#ax=df[item].value\_counts().plot(kind = 'bar')

ax=sns.countplot(item,data = df\_,palette="Pastel1")

plt.xlabel(str(item),fontsize=14)

plt.ylabel('Count',fontsize=14)

plt.xticks(fontsize=13)

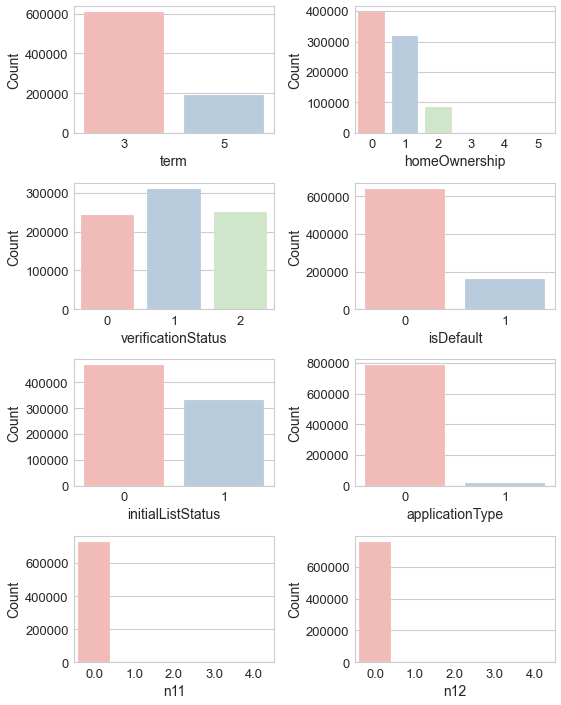
plt.yticks(fontsize=13)

#plt.title("Churn by "+ str(item))

i=i+1

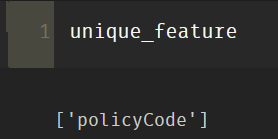
plt.tight\_layout()

plt.show()

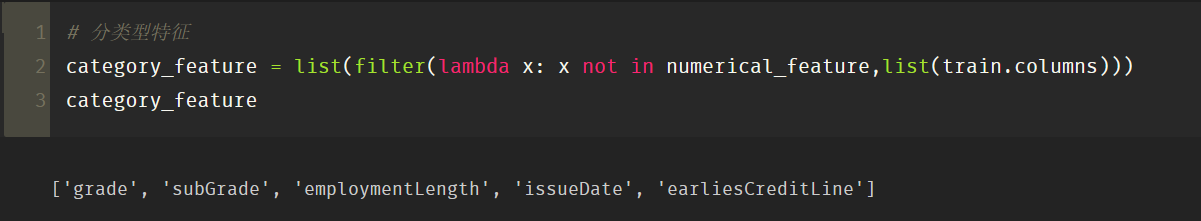


* 对于单值变量

单值变量表示该特征只有一种类别，对于数值全部都一样的特征，可以考虑直接删除。

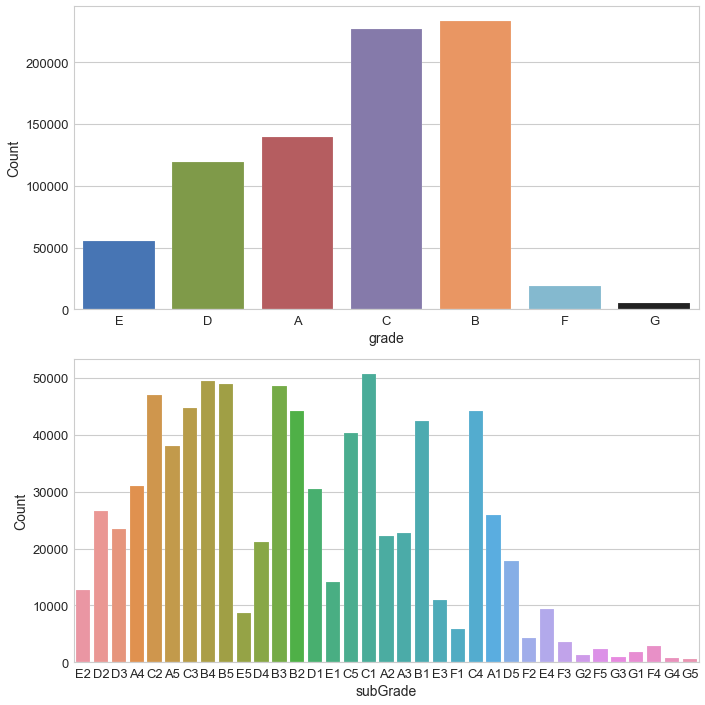


2.2、分类型特征



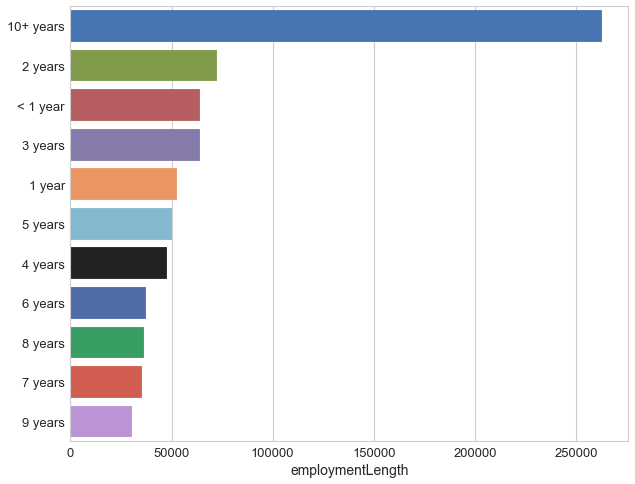
这里 "grade"为贷款等级，"subGrade"为贷款等级之子级，"employmentLength"为就业年限，"issueDate"为贷款发放的月份，"earliesCreditLine"为借款人最早报告的信用额度开立的月份，共有5个分类型特征。

* 分类型特征可视化呈现：



可以看出对于grade特征中A\B\C等级的贷款占比比较大。

* employmentLength就业年限可视化呈现

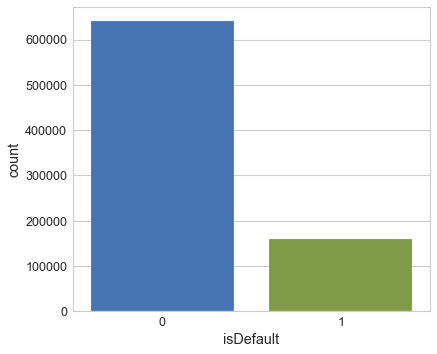


可以看到，就业年限最多是 10+year。

2.3、目标变量（标签y）的分布

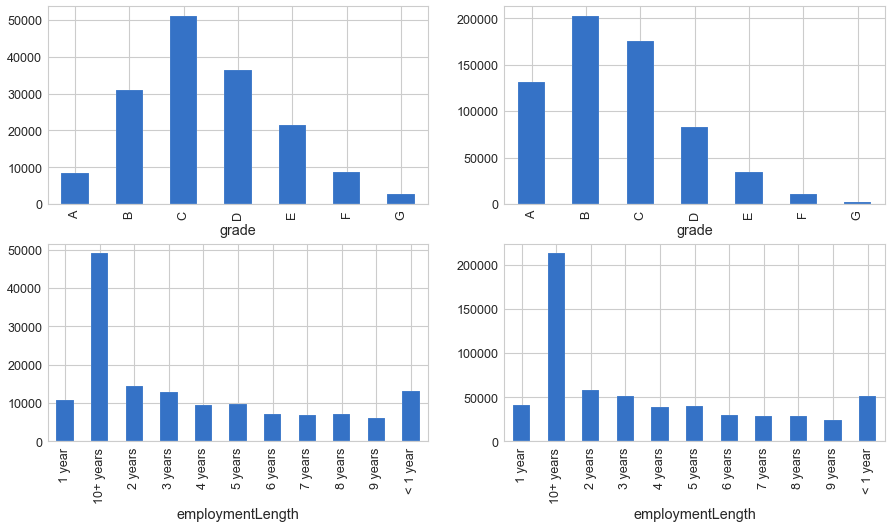
* 查看目标变量（标签）是否平衡。

若分类问题中各类别样本数量差距太大，则会造成样本不均衡的问题。样本不均衡不利于建立与训练出正确的模型，且不能做出合理的评估。



可以看到，贷款违约与不违约的比例大约为1：4，样本较不平衡，这是金融风控模型评估的中常见的现象，大多数的人都是不会拖欠贷款的。

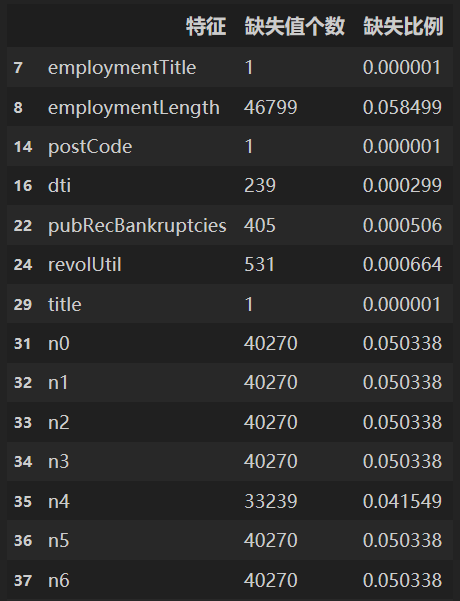
* 目标变量和分类类别之间的分布关系

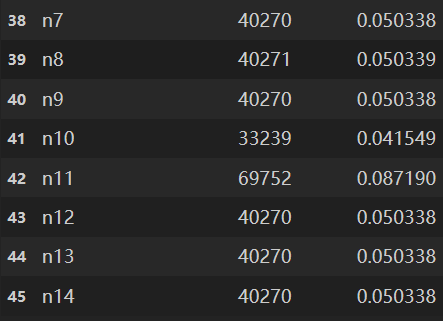


2.4、缺失值查看

如果缺失值过多会对整体的模型结果产生一定的影响，因此每次在建模之前都需要对数据的缺失值情况就行查看，若有缺失情况，需要在后续特征工程中进行填补。

* 缺省值查看

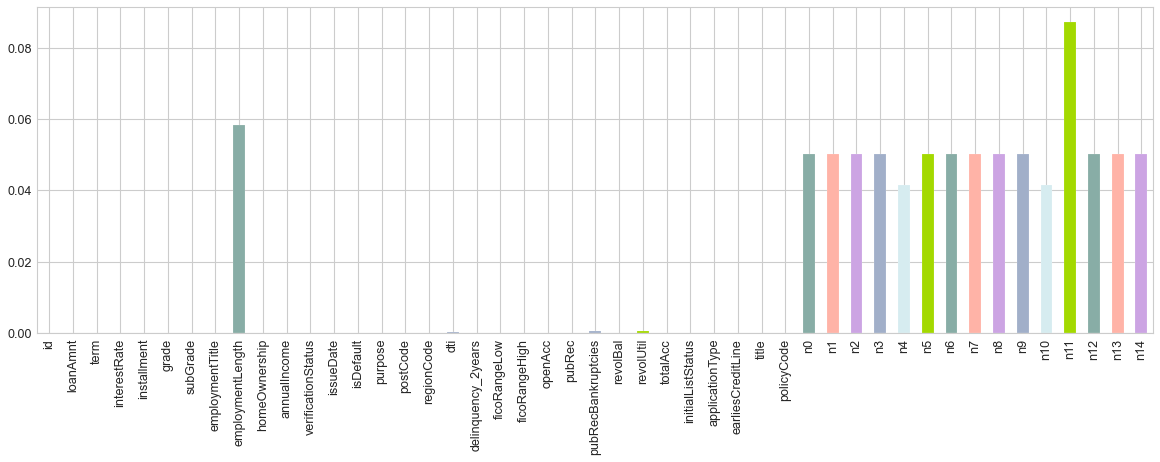




可以看到employmentTitle、employmentLength、dti 以及匿名特征等字段存在缺省值。从上面的结果可以看出train数据集中的47个字段有22个存在缺省值的情况。

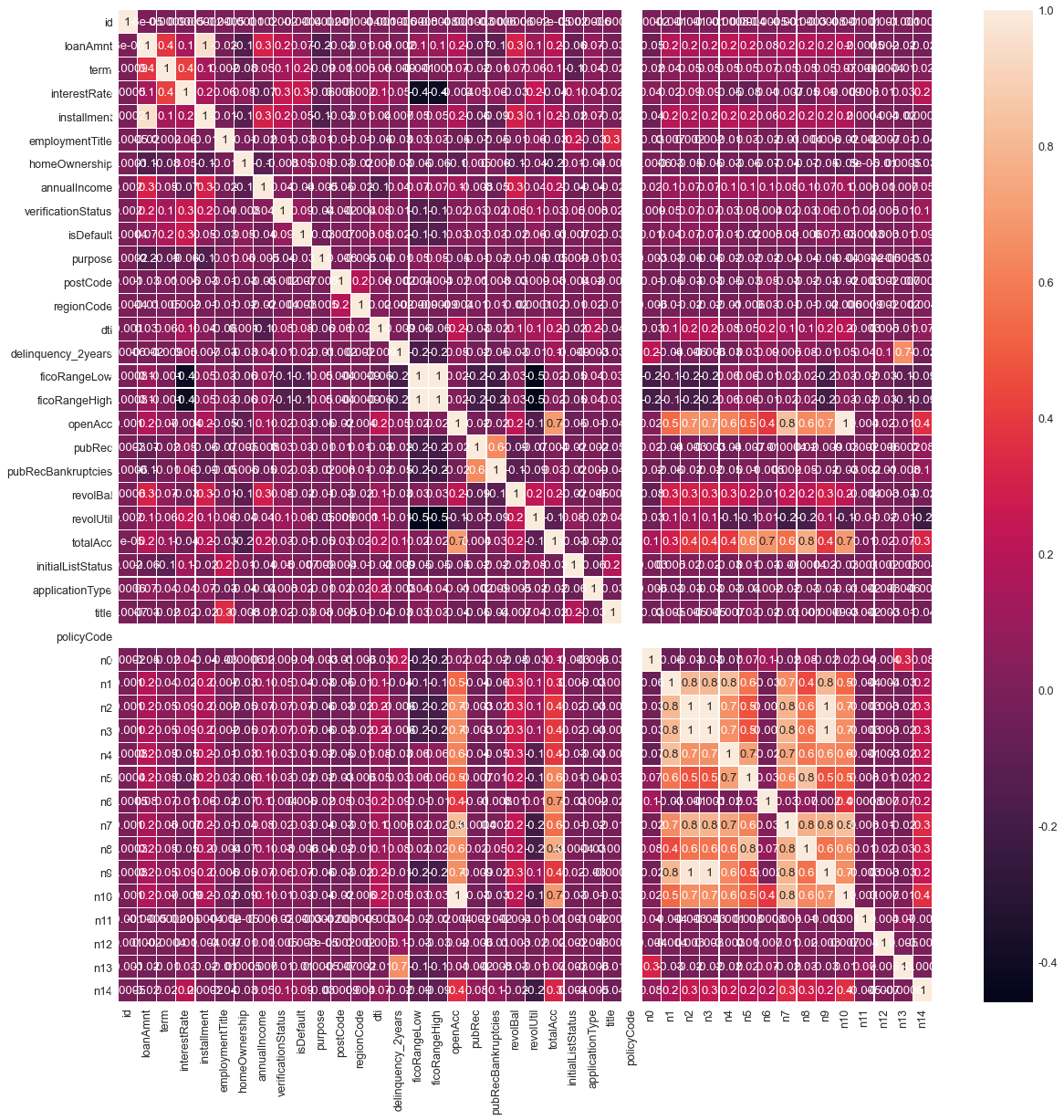
* 可视化一下缺省值数量占比。

一般对于缺失值，需要进行横纵对比。纵向（从列方向）：如果nan存在的过多，说明这一列对label的影响几乎不起作用了，可以考虑删掉。如果缺失值很小一般可以选择填充。比如占到总数的50%，理论上对分析作用不大，这样就可以省略该字段。横向（从行方向）：如果在数据集中，某些样本数据的大部分列都是缺失的且样本足够的情况下可以考虑删除。



可以看到，所有的特征缺失值都在10%以内，这里考虑全部保留。

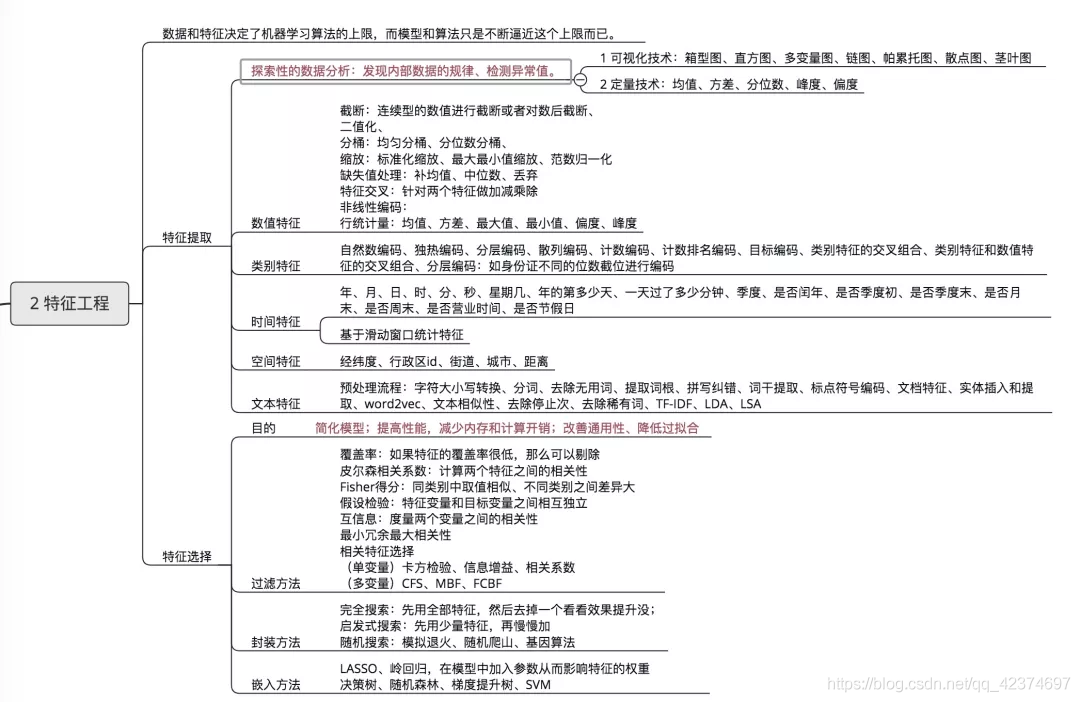
2.5、数据相关关系



可以看到，有些变量之间的相关性还是很强的，比如贷款总额loanAmnt 和分期付款金额installment 相关性为1，ficoRangeLow he ficoRangeHigh 相关性为1…，这种情况后面再特征选择时考虑删除。

**六、特征工程**

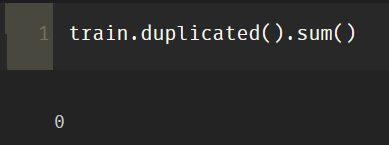
基本的EDA探索完成后，就可以进行特征工程，在数据挖掘中，大部分时间都是在做特征工程特征工程包括数据预处理、缺失值以及异常值的处理、数据分桶处理以及特征交互、编码、选择。



在特征工程中我走了很多弯路，原因就如前面所说的数据处理和特征选择地过于复杂，导致数据过于干净，使得用于预测的特征偏离了实际，导致分数不高。下面我将会从一开始的处理开始描述，将从头至尾地描述出我在这部分进行的具体操作。

* **开始时的工作**

1. **重复值处理**



**2.** **缺失值填补**

用户在登录界面输入用户名和密码并经过系统验证后，如果没有被管理员封禁，则可以进入主服务界面进行相关操作（新用户注册后使用）。

传统地，如果是分类型特征，采用众数进行填补。如果是连续型特征，采用均值进行填补。还要考虑，均值一般适用于近似正态分布数据，观测值较为均匀散布均值周围；中位数一般适用于偏态分布或者有离群点数据，中位数是更好地代表数据中心趋势；众数一般用于类别变量，无大小、先后顺序之分。所以对于连续变量对于数据近似符合正态分布，用该变量的均值填补缺失。对于数据存在偏态分布的情况，采用中位数进行填补。

* 数值型特征用中位数填补：

# 训练集

train[numerical\_feature] = train[numerical\_feature].fillna(train[numerical\_feature].median())

# 测试集

test[numerical\_feature] = test[numerical\_feature].fillna(train[numerical\_feature].median())

* 分类型特征用众数填补：

# 训练集

train[category\_feature] = train[category\_feature].fillna(train[category\_feature].mode())

# 测试集

test[category\_feature] = test[category\_feature].fillna(train[category\_feature].mode())

* 之后employmentLength 列还存在缺失值, 采用决策树来填补

from sklearn.tree import DecisionTreeClassifier

empLenNotNullInd = train.employmentLength.notnull() # 不是空的行，返回True

columns = ['postCode','regionCode','employmentTitle','annualIncome'] # 用四个特征来预测employmentLength

train\_empLen\_X = train.loc[empLenNotNullInd,columns]

train\_empLen\_y = train.employmentLength[empLenNotNullInd]

DTC = DecisionTreeClassifier() # 实例化

DTC.fit(train\_empLen\_X ,train\_empLen\_y) # 训练

print(DTC.score(train\_empLen\_X ,train\_empLen\_y))

# 预测

for data in [train,test]:

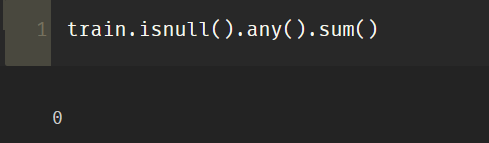
empLenIsNullInd = data.employmentLength.isnull()

test\_empLen\_X = data.loc[empLenIsNullInd,columns]

empLen\_pred = DTC.predict(test\_empLen\_X)

data.employmentLength[empLenIsNullInd] = empLen\_pred

填补完毕:



**3.异常值处理**

异常值的存在很可能会影响模型的最终结果，但是当我们发现异常值的时候也不能马上就删除，应该先看看这个异常值是不是有特殊原因造成的，特别是在金融风控问题中，异常值的出现往往是存在意义的。

如果不是因为特殊原因造成的，可以先观察这个异常值出现的频率。若异常值只出现了一次，多半是输入错误，直接把异常值删除即可；若异常值出现了多次，可以和业务人员沟通，可能这是某种特殊表示，如果是人为造成的错误，留着是没有用，只要数据量不是太大，都可以删除；若异常值占到总数据量的10%以上，不能轻易删除。可以考虑把异常值替换成非异常但是非干扰的项，比如说用0来进行替换，或者把异常当缺失值，用均值或者众数来进行替换。通常，在进行EDA的时候会利用描述统计的方法，查看特征的均值、极大值、极小值等信息，结合常识来判断是否存在异常值。

* 均方差 3σ

在统计学中，如果一个数据分布近似正态，那么大约 68% 的数据值会在均值的一个标准差范围内，大约 95% 会在两个标准差范围内，大约 99.7% 会在三个标准差范围内。

def find\_outliers\_by\_3segama(data,fea):

data\_std = np.std(data[fea])

data\_mean = np.mean(data[fea])

outliers\_cut\_off = data\_std \* 3

lower\_rule = data\_mean - outliers\_cut\_off

upper\_rule = data\_mean + outliers\_cut\_off

data[fea+'\_outliers'] = data[fea].apply(lambda x:str('异常值') if x > upper\_rule or x < lower\_rule else '正常值')

return data

data\_train = train.copy()

for fea in numerical\_feature:

data\_train = find\_outliers\_by\_3segama(data\_train,fea)

print(data\_train[fea+'\_outliers'].value\_counts())

print(data\_train.groupby(fea+'\_outliers')['isDefault'].sum())

print('\*'\*10)

**4.** **时间数据处理**

对于本赛题，时间数据有 issueDate ，可以将其转化成时间格式（issueDateDT特征表示数据日期离数据集中日期最早的日期（2007-06-01）的天数）。

* 训练集时间数据处理

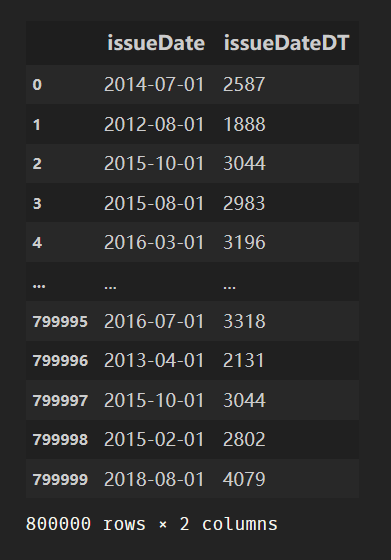
import datetime

# 转化成时间格式 issueDateDT特征表示数据日期离数据集中日期最早的日期（2007-06-01）的天数

train['issueDate'] = pd.to\_datetime(train['issueDate'],format='%Y-%m-%d')

startdate = datetime.datetime.strptime('2007-06-01', '%Y-%m-%d')

train['issueDateDT'] = train['issueDate'].apply(lambda x: x-startdate).dt.days



* 训练集时间数据处理

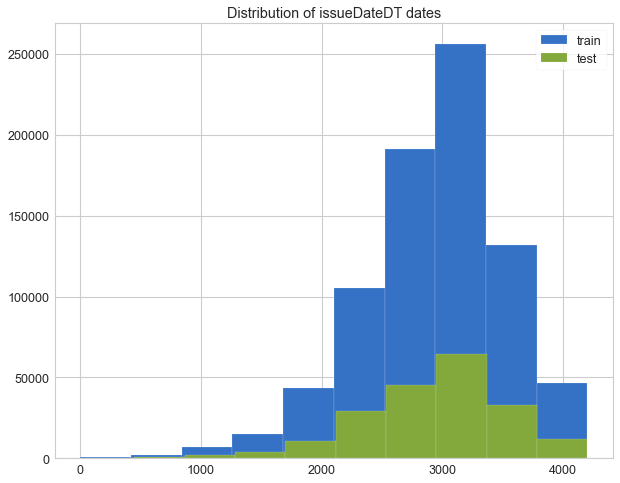
#转化成时间格式

test['issueDate'] = pd.to\_datetime(train['issueDate'],format='%Y-%m-%d')

startdate = datetime.datetime.strptime('2007-06-01', '%Y-%m-%d')

test['issueDateDT'] = test['issueDate'].apply(lambda x: x-startdate).dt.days

* 可视化



**5.特征交叉**

表示特征之间的相互作用，再线性模型中引入非线性性质，提升模型表达能力。

将issueDate贷款发放时的年份减去借款人最早报告的信用额度开立的年份，得到新的特征，即开卡年限CreditLine。

train\_earliesCreditLine\_year = train['earliesCreditLine'].apply(lambda x:x[-4:]).astype('int64')

test\_earliesCreditLine\_year = test['earliesCreditLine'].apply(lambda x:x[-4:]).astype('int64')

train\_issueDate\_year = train['issueDate'].astype('str').apply(lambda x:x[:4]).astype('int64')

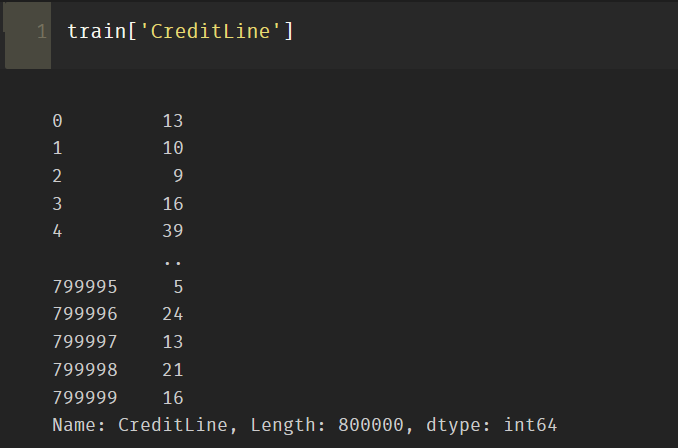
test\_issueDate\_year = test['issueDate'].astype('str').apply(lambda x:x[:4]).astype('int64')

train['CreditLine'] = train\_issueDate\_year - train\_earliesCreditLine\_year

test['CreditLine'] = test\_issueDate\_year - test\_earliesCreditLine\_year

train = train.drop(['earliesCreditLine','issueDate'],axis=1)

test = test.drop(['earliesCreditLine','issueDate'],axis=1)



**6.特征编码**

对类别型特征进行转换，使其变为数值特征。具体有以下几种方法：序号编码：适用于类别间存在大小关系的特征。比如级别高中低，可以对应 321；oneHot 编码：适用于不具有大小关系的特征。比如地名；二进制编码：先给每个类别赋予一个序号 ID，然后对 ID 进行二进制编码，最终得到和 OneHot 类似的 0-1 向量，但是维度更小。

首先将 employmentLength 进行简单的处理，再进行编码

这里将就业年限特征转换为数值（把数字后面的years去掉并且把10+改成10，<1改成0）：

def employmentLength\_to\_int(s):

if pd.isnull(s):

return s

else:

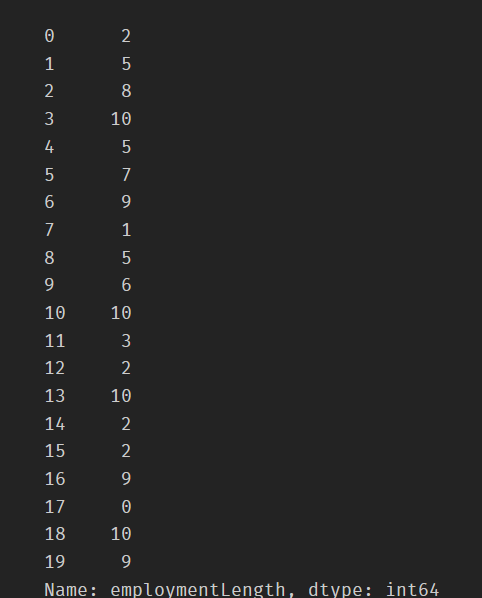
return np.int8(s.split()[0])

for data in [train, test]:

data['employmentLength'].replace(to\_replace='10+ years', value='10 years', inplace=True)

data['employmentLength'].replace('< 1 year', '0 years', inplace=True)

data['employmentLength'] = data['employmentLength'].apply(employmentLength\_to\_int)



* 接下来，对其余分类型特征进行编码，像等级grade、subGrade这种类别特征，虽然是表示类别的数据，但是信用评级是有高低大小之分的，是有优先级的，所以可以直接自映射，转化为数值类型。

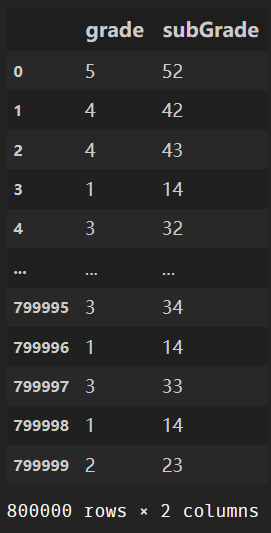
a2z = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'

a2z\_code = np.arange(1,27)

a2z\_mapping = dict(zip(a2z, a2z\_code))

for data in [train,test]:

data.loc[:,['grade','subGrade']] = data.loc[:,['grade','subGrade']].applymap(lambda g:g.replace(g[0], str(a2z.index(g[0])+1))).astype('int')



* 对于离散型特征，可以使用OneHotEncoder独热编码

from sklearn.preprocessing import OneHotEncoder

oh = OneHotEncoder(sparse=False)

oh.fit(train[['homeOwnership','verificationStatus','purpose']])

OneHot1 = oh.transform(train[['homeOwnership','verificationStatus','purpose']])

OneHot2 = oh.transform(test[['homeOwnership','verificationStatus','purpose']])

OneHot1.shape

train = pd.concat([train, pd.DataFrame(OneHot1)], axis=1)

test = pd.concat([test, pd.DataFrame(OneHot2)], axis=1)

train = train.drop(['homeOwnership','verificationStatus','purpose'],axis=1)

test = test.drop(['homeOwnership','verificationStatus','purpose'],axis=1)

train.shape

**7.特征选择**

* 人工判断与目标无关联特征为"id"，需删除

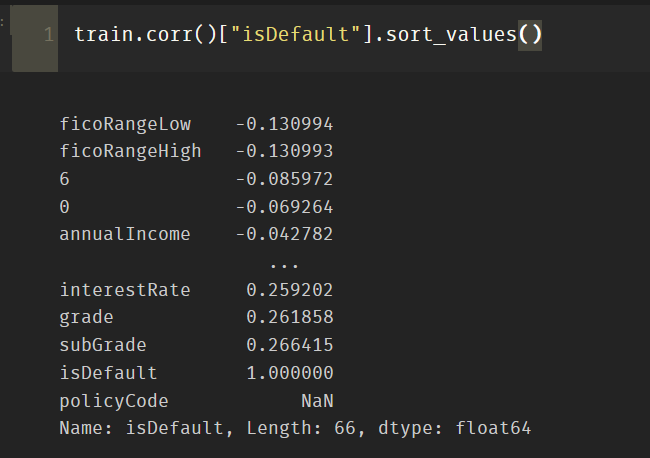
train=train.drop(["id"],axis=1)

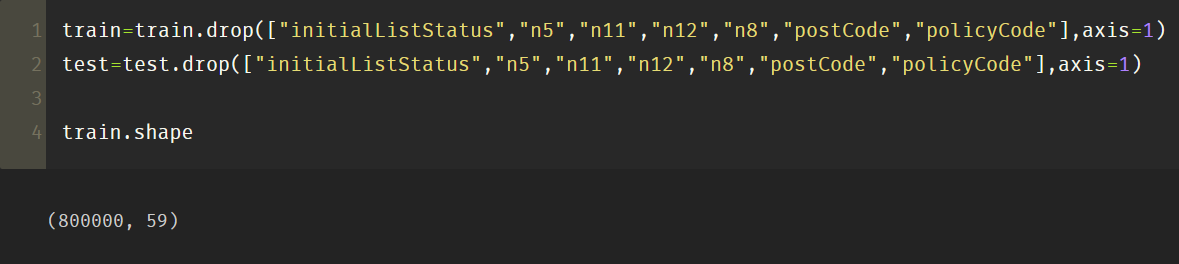
train.shape

test=test.drop(["id"],axis=1)

test.shape

* 求出各个特征与目标的相关系数，综合考虑排除corr小于0.01的特征





* 特征间高相关过滤

# 显示相关性高于0.6的变量

def getHighRelatedFeatureDf(corr\_matrix, corr\_threshold):

highRelatedFeatureDf = pd.DataFrame(corr\_matrix[corr\_matrix>corr\_threshold].stack().reset\_index())

highRelatedFeatureDf.rename({'level\_0':'feature\_x', 'level\_1':'feature\_y', 0:'corr'}, axis=1, inplace=True)

highRelatedFeatureDf = highRelatedFeatureDf[highRelatedFeatureDf.feature\_x != highRelatedFeatureDf.feature\_y]

highRelatedFeatureDf['feature\_pair\_key'] = highRelatedFeatureDf.loc[:,['feature\_x', 'feature\_y']].apply(lambda r:'#'.join(np.sort(r.values)), axis=1)

highRelatedFeatureDf.drop\_duplicates(subset=['feature\_pair\_key'],inplace=True)

highRelatedFeatureDf.drop(['feature\_pair\_key'], axis=1, inplace=True)

return highRelatedFeatureDf

getHighRelatedFeatureDf(train.corr(),0.6)

"loanAmnt"贷款金额，"installment"分期付款金额两个特征间相关系数为0.95；"ficoRangeLow"fico所属的下限范围，"ficoRangeHigh"fico所属的上限范围两个特征间相关系数为1；"openAcc"未结信用额度的数量，“n10” 两个特征间相关系数为0.93；“n3”，"n2"两个特征间相关系数为1；“n3”，“n9” 两个特征间相关系数为0.98。根据高相关特征，综合考虑他们与目标的相关性，删除特征"installment",“ficoRangeHigh”,“openAcc”,“n3”,“n9”。

* 低方差过滤

train.var().sort\_values()

col = ['applicationType']

for data in [train,test]:

data.drop(col,axis=1,inplace=True)

* **最终进行的处理**
* 导入模块和数据：

import pandas as pd

import numpy as np

from sklearn.model\_selection import KFold

from sklearn.metrics import auc, roc\_curve

from lightgbm import LGBMRegressor

from catboost import CatBoostRegressor

from sklearn.preprocessing import StandardScaler

import datetime

np.random.seed(2020)

train\_file\_name = 'G://classStudy/II 2/机器学习/大作业/贷款违约预测/train.csv'

test\_file\_name = 'G://classStudy/II 2/机器学习/大作业/贷款违约预测/testA.csv'

df\_train = pd.read\_csv(train\_file\_name)

df\_test = pd.read\_csv(test\_file\_name)

**1.对训练集进行处理**

* 对grade进行特征编码

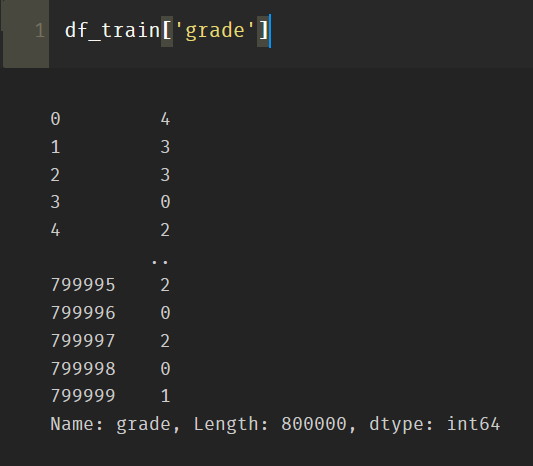
因为grade特征，虽然是表示类别的数据，但是信用评级是有高低大小之分的，是有优先级的，所以也可以使用labelencode编码。

from sklearn.preprocessing import LabelEncoder

labelEncoder = LabelEncoder()

df\_train['grade'] = labelEncoder.fit\_transform(df\_train['grade'])

df\_train['grade'] = labelEncoder.fit\_transform(df\_train['grade'])



* 对employmentLength进行自然映射，将其从分类型变为数值型

employmentLength = ['< 1 year','1 year','2 years',

'3 years', '4 years', '5 years',

'6 years', '8 years', '7 years','9 years','10+ years']

j = 0

for i in employmentLength:

df\_train['employmentLength'] = df\_train['employmentLength'].replace(i, j)

j += 1

* 对subGrade与目标进行特征交互和特征编码

\_ = pd.crosstab(df\_train.subGrade, df\_train.isDefault)

\_["yp"] = \_[1]/(\_[0]+\_[1])

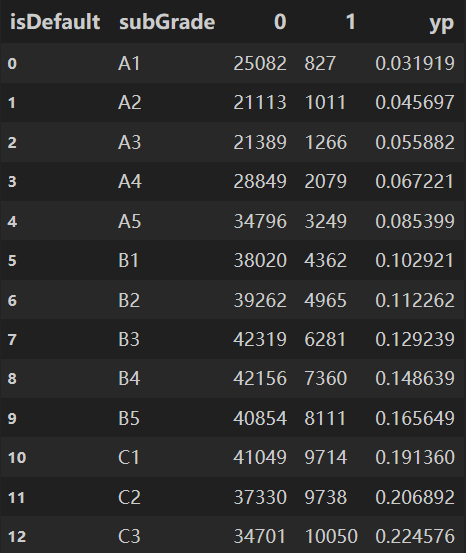
\_.reset\_index(inplace=True)

\_.sort\_values(by="yp", inplace=True)

df\_train = pd.merge(df\_train, \_[["subGrade", "yp"]], on="subGrade", how="left")

df\_train['subGrade'] = labelEncoder.fit\_transform(df\_train['subGrade'])

“yp”为不同subGrade中正例所占的比例，能够帮助后面更好地训练模型



* 对于issueDate、issueDateDT、earliesCreditLine等时间信息处理

df\_train['issueDate'] = pd.to\_datetime(df\_train['issueDate'],format='%Y-%m-%d')

startdate = datetime.datetime.strptime('2007-06-01', '%Y-%m-%d')

df\_train['issueDateDT'] = df\_train['issueDate'].apply(lambda x: x-startdate).dt.days

df\_train['earliesCreditLine'] = df\_train['earliesCreditLine'].apply(lambda s: int(s[-4:]))

‘issueDate’使用年月日的格式,‘issueDateDT’为数据时间减去最早时间，‘earlierCreditLine’保留年份。在实际生活中，借款的时间是衡量违约的一重要因素，所以时间这一特征很重要，在一开始的特征工程中，我未保留月日的信息，从而导致预测结果不好。

1. **对于测试集进行和训练集一样的特征工程处理**

df\_test['grade'] = labelEncoder.fit\_transform(df\_test['grade'])

employmentLength = ['< 1 year','1 year','2 years',

'3 years', '5 years', '4 years',

'6 years', '8 years', '7 years','9 years','10+ years']

j = 0

for i in employmentLength:

df\_test['employmentLength'] = df\_test['employmentLength'].replace(i, j)

j += 1

df\_test = pd.merge(df\_test, \_[["subGrade", "yp"]], on="subGrade", how="left")

df\_test['subGrade'] = labelEncoder.fit\_transform(df\_test['subGrade'])

df\_test['issueDate'] = pd.to\_datetime(df\_test['issueDate'],format='%Y-%m-%d')

startdate = datetime.datetime.strptime('2007-06-01', '%Y-%m-%d')

df\_test['issueDateDT'] = df\_test['issueDate'].apply(lambda x: x-startdate).dt.days

df\_test['earliesCreditLine'] = df\_test['earliesCreditLine'].apply(lambda s: int(s[-4:]))

**七、建模分析**

在完成相关的特征处理后，接下来进行建模分析，通过调节参数得到性能更强的模型,再进行模型融合，模型融合是比赛后期上分的重要手段，模型融合后结果会有大幅提升。

1. **模型考虑**

* Xgboost

GBDT算法只利用了一阶的导数信息，xgboost对损失函数做了二阶的泰勒展开，并在目标函数之外加入了正则项对整体求最优解，用以权衡目标函数的下降和模型的复杂程度，避免过拟合。xgboost对应的模型就是一堆CART树。主要思想就是弱分类器，一起组合成一个强分类器。

一开始树是0，然后往里面加树，相当于多了一个函数，再加第二棵树，相当于又多了一个函数…等等，这里需要保证加入新的函数能够提升整体对表达效果。提升表达效果的意思就是说加上新的树之后，目标函数的值会下降。如果叶子结点的个数太多，那么过拟合的风险会越大，所以这里要限制叶子结点的个数，所以在原来目标函数里要加上一个惩罚项。

Obj代表了当我们指定一个树的结构的时候，在目标上最多会减少多少，我们可以把它叫做结构分数，这个分数越小越好。

xgboost的优势：正则化——减少过拟合;并行处理——提升速度;高度灵活性——允许用户定义自定义优化目标和评价标准;缺失值处理——内置处理缺失值的规则;内置交叉验证XGBoost允许在每一轮boosting迭代中使用交叉验证;在已有的模型基础上继续——可以在上一轮的结果上继续训练;剪枝——XGBoost会一直分裂到指定的最大深，然后回过头来剪枝。如果某个节点之后不再有正值，它会去除这个分裂。

* Lightgbm

Xgboost通过预排序能精确地找到分割点。但是缺点也很明显：空间消耗大、时间上也有较大的开销，在遍历每一个分割点的时候，都需要进行分裂增益的计算，消耗的代价大。

LightGBM是一个实现GBDT算法的框架，支持高效率的并行训练，并且具有更快的训练速度、更低的内存消耗、更好的准确率、支持分布式可以快速处理海量数据等优点。 LightGBM提出的主要原因就是为了解决GBDT在海量数据遇到的问题。本题120万条数据就比较符合这一点。

优点：速度更快——采用了直方图算法将遍历样本转变为遍历直方图，极大的降低了时间复杂度；在训练过程中采用单边梯度算法过滤掉梯度小的样本，减少了大量的计算；采用了基于 Leaf-wise 算法的增长策略构建树，减少了很多不必要的计算量；采用优化后的特征并行、数据并行方法加速计算，当数据量非常大的时候还可以采用投票并行的策略；对缓存也进行了优化，增加了缓存命中率；内存更小——采用了直方图算法将存储特征值转变为存储 bin 值，降低了内存消耗；在训练过程中采用互斥特征捆绑算法减少了特征数量，降低了内存消耗。

* Catboost

CatBoost是一种基于对称决策树为基学习器实现的参数较少、支持类别型变量和高准确性的GBDT框架，主要解决的痛点是高效合理地处理类别型特征，这一点从它的名字中可以看出来，CatBoost是由Categorical和Boosting组成。此外，CatBoost还解决了梯度偏差以及预测偏移的问题，从而减少过拟合的发生，进而提高算法的准确性和泛化能力。

与XGBoost、LightGBM相比，CatBoost的创新点有:嵌入了自动将类别型特征处理为数值型特征的创新算法。首先对计算某个类别特征出现的频率，之后加上超参数，生成新的数值型特征。这一点非常适合本实验的这些特征，这使得不用对于分类特征做过多的处理，从而避免了数据处理地过于干净。Catboost还使用了组合类别特征，可以利用到特征之间的联系，这极大的丰富了特征维度。采用排序提升的方法对抗训练集中的噪声点，从而避免梯度估计的偏差，进而解决预测偏移的问题。

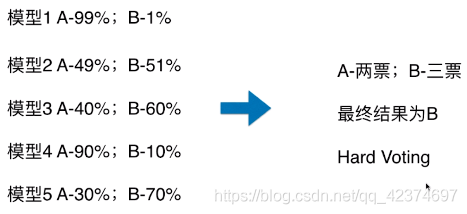
其有点为：性能卓越——在性能方面可以匹敌任何先进的机器学习算法；鲁棒性——它减少了对很多超参数调优的需求，并降低了过度拟合的机会，这也使得模型变得更加具有通用性；易于使用——提供与scikit集成的Python接口，以及R和命令行界面；实用——可以处理类别型、数值型特征；可扩展——支持自定义损失函数。

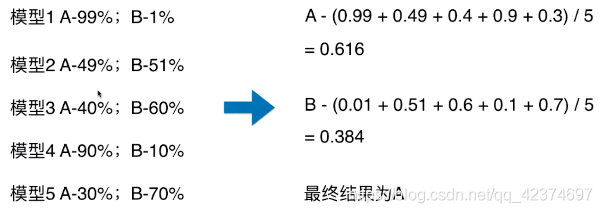
* 模型融合之stacking

stacking构建多层模型，并利用预测结果再拟合预测。Stacking将若干基学习器获得的预测结果作为新的训练集训练一个学习器。但是由于直接由多个基学习器获得结果直接带入模型中，容易导致过拟合。所以在使用多个基模型进行预测时，可以考虑K折验证，防止过拟合。Stacking中由于两层使用的数据不同，所以可以避免信息泄露的问题。

* 模型融合之Voting

Voting针对分类问题，按照少数服从多数的方法进行投票，比较符合本题。





1. **尝试过程**

* **使用开始时处理后的数据进行建模**
* Lightbgm

X = train.drop(['isDefault'], axis=1)

y = train.loc[:,'isDefault']

kf = KFold(n\_splits=5, shuffle=True, random\_state=525)

X\_train\_split, X\_val, y\_train\_split, y\_val = train\_test\_split(X, y, test\_size=0.2)

import lightgbm as lgb

cv\_scores = []

for i, (train\_index, val\_index) in enumerate(kf.split(X, y)):

X\_train, y\_train, X\_val, y\_val = X.iloc[train\_index], y.iloc[train\_index], X.iloc[val\_index], y.iloc[val\_index]

train\_matrix = lgb.Dataset(X\_train, label=y\_train)

valid\_matrix = lgb.Dataset(X\_val, label=y\_val)

params = {

'boosting\_type': 'gbdt',

'objective': 'binary',

'learning\_rate': 0.1,

'metric': 'auc',

'min\_child\_weight': 1e-3,

'num\_leaves': 31,

'max\_depth': -1,

'seed': 525,

'nthread': 8,

'silent': True,

}

model = lgb.train(params, train\_set=train\_matrix, num\_boost\_round=20000, valid\_sets=valid\_matrix, verbose\_eval=1000, early\_stopping\_rounds=200)

val\_pred = model.predict(X\_val, num\_iteration=model.best\_iteration)

cv\_scores.append(roc\_auc\_score(y\_val, val\_pred))

print(cv\_scores)

print("lgb\_scotrainre\_list:{}".format(cv\_scores))

print("lgb\_score\_mean:{}".format(np.mean(cv\_scores)))

print("lgb\_score\_std:{}".format(np.std(cv\_scores)))

from sklearn import metrics

from sklearn.metrics import roc\_auc\_score

al\_pre\_lgb = model.predict(X\_val, num\_iteration=model.best\_iteration)

fpr, tpr, threshold = metrics.roc\_curve(y\_val, val\_pred)

roc\_auc = metrics.auc(fpr, tpr)

print('AUC：{}'.format(roc\_auc))

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

plt.title('Validation ROC')

plt.plot(fpr, tpr, 'b', label = 'Val AUC = %0.4f' % roc\_auc)

plt.ylim(0,1)

plt.xlim(0,1)

plt.legend(loc='best')

plt.title('ROC')

plt.ylabel('True Positive Rate')

plt.xlabel('False Positive Rate')

# 画出对角线

plt.plot([0,1],[0,1],'r--')

plt.show()

本机AUC得分为0.7338

* Xgboost

X = train.drop(['isDefault'], axis=1)

y = train.loc[:,'isDefault']

Xtrain, Xtest, Ytrain, Ytest = train\_test\_split(X,y,test\_size=0.3)

from xgboost.sklearn import XGBClassifier

clf1 = XGBClassifier(n\_jobs=-1)

clf1.fit(Xtrain,Ytrain)

clf1.score(Xtest,Ytest)

from sklearn.metrics import roc\_curve, auc

predict\_proba = clf1.predict\_proba(Xtest)

false\_positive\_rate, true\_positive\_rate, thresholds = roc\_curve(Ytest, predict\_proba[:,1])

auc(false\_positive\_rate, true\_positive\_rate)

本机计算模型结构的AUC面积0.7326304866618416

进行网格调参找更好的参数

from xgboost.sklearn import XGBClassifier

from sklearn.model\_selection import GridSearchCV

# 其余参数

other\_params = {'learning\_rate': 0.1,

'n\_estimators': 100,

'max\_depth': 5,

'min\_child\_weight': 1,

'seed': 0,

'subsample': 0.8,

'colsample\_bytree': 0.8,

'gamma': 0,

'reg\_alpha': 0,

'reg\_lambda': 1}

# 待调参数

param\_test1 = {

'max\_depth':list(range(4,9,2)),

'min\_child\_weight':list(range(1,6,2))

}

xgb1 = XGBClassifier(\*\*other\_params)

# 网格搜索

gs1 = GridSearchCV(xgb1,param\_test1,cv = 5,scoring = 'roc\_auc',n\_jobs = -1,verbose=2)

best\_model1=gs1.fit(Xtrain,Ytrain)

print('最优参数：',best\_model1.best\_params\_)

print('最佳模型得分：',best\_model1.best\_score\_)

other\_params = {'learning\_rate': 0.1,

'n\_estimators': 100,

'max\_depth': 4,

'min\_child\_weight': 5,

'seed': 0,

'subsample': 0.8,

'colsample\_bytree': 0.8,

'gamma': 0,

'reg\_alpha': 0,

'reg\_lambda': 1}

param\_test = {

'gaama':[0,0.05,0.1,0.2,0.3]

}

xgb = XGBClassifier(\*\*other\_params)

gs = GridSearchCV(xgb,param\_test,cv = 5,scoring = 'roc\_auc',n\_jobs = -1,verbose=2)

best\_model=gs.fit(Xtrain,Ytrain)

print('最优参数：',best\_model.best\_params\_)

print('最佳模型得分：',best\_model.best\_score\_)

other\_params = {'learning\_rate': 0.1, 'n\_estimators': 100, 'max\_depth': 4, 'min\_child\_weight': 5, 'seed': 0,

'subsample': 0.8, 'colsample\_bytree': 0.8, 'gamma': 0, 'reg\_alpha': 0, 'reg\_lambda': 1}

param\_test = {

'subsample':[0.6,0.7,0.8,0.9],

'colsample\_bytree':[0.6,0.7,0.8,0.9]

}

xgb = XGBClassifier(\*\*other\_params)

gs = GridSearchCV(xgb,param\_test,cv = 5,scoring = 'roc\_auc',n\_jobs = -1,verbose=2)

best\_model=gs.fit(Xtrain,Ytrain)

print('最优参数：',best\_model.best\_params\_)

print('最佳模型得分：',best\_model.best\_score\_)

other\_params = {'learning\_rate': 0.1, 'n\_estimators': 100, 'max\_depth': 4, 'min\_child\_weight': 5, 'seed': 0,

'subsample': 0.7, 'colsample\_bytree': 0.7, 'gamma': 0, 'reg\_alpha': 0, 'reg\_lambda': 1}

param\_test = {

'reg\_alpha': [4,5,6,7],

'reg\_lambda': [0,0.01,0.05, 0.1]

}

xgb = XGBClassifier(\*\*other\_params)

gs = GridSearchCV(xgb,param\_test,cv = 5,scoring = 'roc\_auc',n\_jobs = -1,verbose=2)

best\_model=gs.fit(Xtrain,Ytrain)

print('最优参数：',best\_model.best\_params\_)

print('最佳模型得分：',best\_model.best\_score\_)

other\_params = {'learning\_rate': 0.1, 'n\_estimators': 100, 'max\_depth': 4, 'min\_child\_weight': 5, 'seed': 0,

'subsample': 0.7, 'colsample\_bytree': 0.7, 'gamma': 0, 'reg\_alpha': 5, 'reg\_lambda': 0.01}

param\_test = {

'learning\_rate': [0.01, 0.05, 0.07, 0.1, 0.2],

'n\_estimators': [100,200,300,400,500]

}

xgb = XGBClassifier(\*\*other\_params)

gs = GridSearchCV(xgb,param\_test,cv = 5,scoring = 'roc\_auc',n\_jobs = -1,verbose=2)

best\_model=gs.fit(Xtrain,Ytrain)

print('最优参数：',best\_model.best\_params\_)

print('最佳模型得分：',best\_model.best\_score\_)

通过调参后的模型

from xgboost.sklearn import XGBClassifier

clf = XGBClassifier(

learning\_rate= 0.05,

n\_estimators= 400,

max\_depth= 4,

min\_child\_weight= 5,

seed= 0,

subsample= 0.7,

colsample\_bytree= 0.7,

gamma= 0,

reg\_alpha= 5,

reg\_lambda=0.01,

n\_jobs = -1)

clf.fit(Xtrain,Ytrain)

clf.score(Xtest,Ytest)

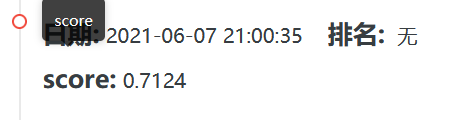
from sklearn.metrics import roc\_curve, auc

predict\_proba = clf.predict\_proba(Xtest)

false\_positive\_rate, true\_positive\_rate, thresholds = roc\_curve(Ytest, predict\_proba[:,1])

auc(false\_positive\_rate, true\_positive\_rate)

本机AUC面积为0.74512067（本机测试很高，但是线上测试只有0.71，模型融合之后同样提升不高）



* Stacking（模型融合）

之后使用之前的训练的lgb和xgb模型作为基分类器，逻辑回归作为目标分类器做stacking。

from mlxtend.classifier import StackingClassifier

gra=GradientBoostingClassifier()

xgb=XGBClassifier()

lgb=LGBMClassifier()

lr = LogisticRegression()

sclf = StackingClassifier(classifiers=[gra, xgb, lgb],

use\_probas=True,

meta\_classifier=lr)

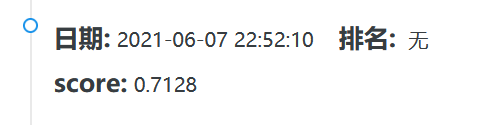
sclf.fit(Xtrain,Ytrain)

pre =sclf.predict\_proba(Xtest)[:,1]

fpr, tpr, thresholds = roc\_curve(Ytest, pre)

score = auc(fpr, tpr)

print(score)



模型融合之后效果也不好，分析出的原因就是前面讨论过的数据太干净的问题，特征偏离了实际，如时间只保留了年份，但是实际违约需要更精确的时间。

* **改变数据处理方式后的建模**

使用特征工程后的特征进行建模：

tags = ['loanAmnt', 'term', 'interestRate', 'installment', 'grade',

'subGrade', 'employmentTitle', 'employmentLength', 'homeOwnership',

'annualIncome', 'verificationStatus', 'issueDateDT', 'earliesCreditLine',

'purpose', 'postCode', 'regionCode', 'dti', 'delinquency\_2years',

'ficoRangeLow', 'ficoRangeHigh', 'openAcc', 'pubRec',

'pubRecBankruptcies', 'revolBal', 'revolUtil', 'totalAcc',

'initialListStatus', 'applicationType', 'title',

'n0', 'n1', 'n2', 'n4', 'n5', 'n6', 'n7', 'n8',

'n9', 'n10', 'n11', 'n12', 'n13', 'n14','yp']

* Lightgbm

lgbr = LGBMRegressor(num\_leaves=30

,max\_depth=10

,learning\_rate=0.01

,n\_estimators=13000

,subsample\_for\_bin=5000

,min\_child\_samples=200

,colsample\_bytree=.2

,reg\_alpha=.1

,reg\_lambda=.1

,seed=2020

)

参数由网格调参得出

* Catboost

cat = CatBoostRegressor(depth=9,

l2\_leaf\_reg=1,

learning\_rate=0.01,

eval\_metric = 'AUC' ,

border\_count = 128,

bagging\_temperature = 0.9 ,

n\_estimators=16000,

early\_stopping\_rounds=500,

subsample = 0.9,

random\_seed=1,

verbose = 0)

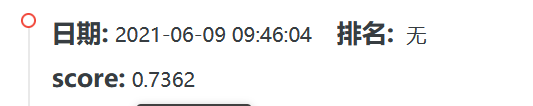
参数由网格调参得出

* Voting（模型融合）

from sklearn.ensemble import VotingRegressor

rg\_model = VotingRegressor([('lgb', lgbr), ('catboost', cat)],n\_jobs=12)

使用前面训练出来的两个模型进行建模，这一次分数提升很多。由于能力和时间问题也就没有继续改进。



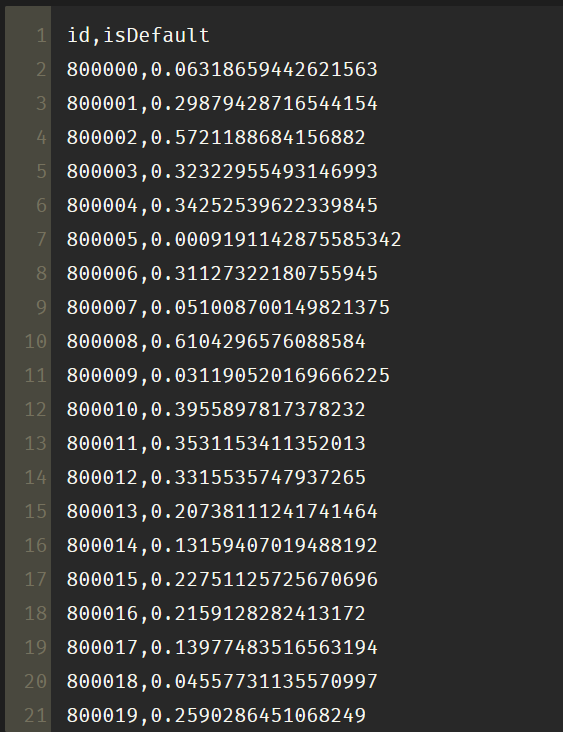
**八、结果**

**1.通过最后得出的模型，将测试集带进去训练，并得到最后的提交文件。**

pre = pd.DataFrame(rg\_model.predict(x\_),columns=['isDefault'])

results = pd.concat([df\_test['id'],pre],axis = 1)

results.to\_csv('submit.csv', index=False)



**2.线上提交文件后，得到成绩**



**九、总结与收获**

通过大作业学习到了一个机器学习项目的完整过程，而不是平时实验时对每个部分分开处理，所以会感觉一个完整的项目想要做好需要很多的精力和知识。从开始数据处理的顺利到建模时分数死活提升不了，还有中间调参时的等待，感觉已经不是单纯的大作业了，更像是对我耐心的考验。做完完整的项目后，感觉还是很有成就感的，特别是看见分数提升之后，感觉还是很好的。

这个项目还教会了我，知识和实践还是有很大不同的，就像是学习的数据处理和特征工程的很多方法在该项目中不太实用，实用之后还会导致结果不理想，原因就是该实验是实际的贷款平台得到的数据，所以那些特征很多不需要处理的，过渡处理反而导致过拟合。

总之，收获很多，期望不会挂科！！！

**十、代码**

import warnings

warnings.filterwarnings('ignore')

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

import datetime

import statsmodels.formula.api as smf

from sklearn.preprocessing import LabelEncoder

from sklearn.model\_selection import train\_test\_split

from sklearn.feature\_selection import chi2

from sklearn.preprocessing import MinMaxScaler

from lightgbm import LGBMClassifier

from sklearn.feature\_selection import SelectKBest

from sklearn.model\_selection import StratifiedKFold,KFold

from sklearn.model\_selection import cross\_val\_score

import xgboost as xgb

import lightgbm as lgb

from catboost import CatBoostRegressor

from sklearn.ensemble import GradientBoostingClassifier

from lightgbm import LGBMRegressor

from catboost import CatBoostRegressor

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import roc\_auc\_score,accuracy\_score,f1\_score,log\_loss

plt.rcParams["font.sans-serif"]=["SimHei"]#正常显示中文

plt.rcParams["axes.unicode\_minus"]=False#正常显示负号

from jupyterthemes import jtplot

jtplot.style(theme='onedork') #选择一个绘图主题

train=pd.read\_csv('G://classStudy/II 2/机器学习/大作业/贷款违约预测/train.csv')

test=pd.read\_csv('G://classStudy/II 2/机器学习/大作业/贷款违约预测/testA.csv')

train.head()

train.shape

test.shape

train.columns

train.info()

train.describe()

# 数值类型

numerical\_feature = list(train.select\_dtypes(exclude=['object']).columns)

numerical\_feature

len(numerical\_feature)

# 连续型变量

serial\_feature = []

# 离散型变量

discrete\_feature = []

# 单值变量

unique\_feature = []

for fea in numerical\_feature:

temp = train[fea].nunique()# 返回的是唯一值的个数

if temp == 1:

unique\_feature.append(fea)

# 自定义变量的值的取值个数小于10就为离散型变量

elif temp <= 10:

discrete\_feature.append(fea)

else:

serial\_feature.append(fea)

serial\_feature

plt.figure(1 , figsize = (8 , 5))

sns.distplot(train.loanAmnt,bins=40)

plt.xlabel('loanAmnt')

label=train.isDefault

label.value\_counts()/len(label)

sns.kdeplot(train.loanAmnt[label[label==1].index], label='1', shade=True)#违约

sns.kdeplot(train.loanAmnt[label[label==0].index], label='0', shade=True)#没有违约

plt.xlabel('loanAmnt')

plt.ylabel('Density');

plt.figure(1 , figsize = (8 , 5))

sns.distplot(train['annualIncome'])

plt.xlabel('annualIncome')

discrete\_feature

for f in discrete\_feature:

print(f, '类型数：', train[f].nunique())

df\_ = train[discrete\_feature]

sns.set\_style("whitegrid") # 使用whitegrid主题

fig,axes=plt.subplots(nrows=4,ncols=2,figsize=(8,10))

for i, item in enumerate(df\_):

plt.subplot(4,2,(i+1))

#ax=df[item].value\_counts().plot(kind = 'bar')

ax=sns.countplot(item,data = df\_,palette="Pastel1")

plt.xlabel(str(item),fontsize=14)

plt.ylabel('Count',fontsize=14)

plt.xticks(fontsize=13)

plt.yticks(fontsize=13)

#plt.title("Churn by "+ str(item))

i=i+1

plt.tight\_layout()

plt.show()

unique\_feature

# 分类型特征

category\_feature = list(filter(lambda x: x not in numerical\_feature,list(train.columns)))

category\_feature

train[category\_feature]

df\_category = train[['grade', 'subGrade']]

sns.set\_style("whitegrid") # 使用whitegrid主题

color = sns.color\_palette()

fig,axes=plt.subplots(nrows=2,ncols=1,figsize=(10,10))

for i, item in enumerate(df\_category):

plt.subplot(2,1,(i+1))

#ax=df[item].value\_counts().plot(kind = 'bar')

ax=sns.countplot(item,data = df\_category)

plt.xlabel(str(item),fontsize=14)

plt.ylabel('Count',fontsize=14)

plt.xticks(fontsize=13)

plt.yticks(fontsize=13)

#plt.title("Churn by "+ str(item))

i=i+1

plt.tight\_layout()

plt.show()

plt.figure(1 , figsize = (10 , 8))

sns.barplot(train["employmentLength"].value\_counts(dropna=False),

train["employmentLength"].value\_counts(dropna=False).keys())

plt.xticks(fontsize=13)

plt.yticks(fontsize=13)

plt.xlabel('employmentLength',fontsize=14)

plt.show()

for i in train[['issueDate', 'earliesCreditLine']]:

print(train[i].value\_counts())

print()

label=train.isDefault

label.value\_counts()/len(label)

sns.countplot(label)

train\_loan\_fr = train.loc[train['isDefault'] == 1]

train\_loan\_nofr = train.loc[train['isDefault'] == 0]

fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2, figsize=(15, 8))

# 目标变量为1时候grade的分布

train\_loan\_fr.groupby("grade").size().plot.bar(ax=ax1)

# 目标变量为0时候grade的分布

train\_loan\_nofr.groupby("grade")["grade"].count().plot.bar(ax=ax2)

# 目标变量为1时候employmentLength的分布

train\_loan\_fr.groupby("employmentLength").size().plot.bar(ax=ax3)

# 目标变量为0时候employmentLength的分布

train\_loan\_nofr.groupby("employmentLength")["employmentLength"].count().plot.bar(ax=ax4)

plt.xticks(rotation=90);

train\_positve = train[train['isDefault'] == 1]

train\_negative = train[train['isDefault'] != 1]

f, ax = plt.subplots(len(numerical\_feature),2,figsize = (10,80))

for i,col in enumerate(numerical\_feature):

sns.distplot(train\_positve[col],ax = ax[i,0],color = "blue")

ax[i,0].set\_title("positive")

sns.distplot(train\_negative[col],ax = ax[i,1],color = 'red')

ax[i,1].set\_title("negative")

plt.subplots\_adjust(hspace = 1)

# 去掉标签

X\_missing = train.drop(['isDefault'],axis=1)

# 查看缺失情况

missing = X\_missing.isna().sum()

missing = pd.DataFrame(data={'特征': missing.index,'缺失值个数':missing.values})

#通过~取反，选取不包含数字0的行

missing = missing[~missing['缺失值个数'].isin([0])]

# 缺失比例

missing['缺失比例'] = missing['缺失值个数']/X\_missing.shape[0]

missing

# 可视化

(train.isnull().sum()/len(train)).plot.bar(figsize = (20,6),color=['#d6ecf0','#a3d900','#88ada6','#ffb3a7','#cca4e3','#a1afc9'])

f, ax = plt.subplots(1,1, figsize = (20,20))

cor = train[numerical\_feature].corr()

sns.heatmap(cor, annot = True, linewidth = 0.2, linecolor = "white", ax = ax, fmt =".1g" )

import pandas as pd

import numpy as np

from sklearn.model\_selection import KFold

from sklearn.metrics import auc, roc\_curve

from lightgbm import LGBMRegressor

from catboost import CatBoostRegressor

from sklearn.preprocessing import StandardScaler

import datetime

np.random.seed(2020)

train\_file\_name = 'G://classStudy/II 2/机器学习/大作业/贷款违约预测/train.csv'

test\_file\_name = 'G://classStudy/II 2/机器学习/大作业/贷款违约预测/testA.csv'

df\_train = pd.read\_csv(train\_file\_name)

df\_test = pd.read\_csv(test\_file\_name)

from sklearn.preprocessing import LabelEncoder

labelEncoder = LabelEncoder()

df\_train['grade'] = labelEncoder.fit\_transform(df\_train['grade'])

df\_train['grade'] = labelEncoder.fit\_transform(df\_train['grade'])

df\_train['grade']

df\_train['employmentLength']

employmentLength = ['< 1 year','1 year','2 years',

'3 years', '4 years', '5 years',

'6 years', '8 years', '7 years','9 years','10+ years']

j = 0

for i in employmentLength:

df\_train['employmentLength'] = df\_train['employmentLength'].replace(i, j)

j += 1

\_ = pd.crosstab(df\_train.subGrade, df\_train.isDefault)

\_["yp"] = \_[1]/(\_[0]+\_[1])

\_.reset\_index(inplace=True)

\_.sort\_values(by="yp", inplace=True)

\_

df\_train = pd.merge(df\_train, \_[["subGrade", "yp"]], on="subGrade", how="left")

df\_train['subGrade'] = labelEncoder.fit\_transform(df\_train['subGrade'])

df\_train['issueDate'] = pd.to\_datetime(df\_train['issueDate'],format='%Y-%m-%d')

startdate = datetime.datetime.strptime('2007-06-01', '%Y-%m-%d')

df\_train['issueDateDT'] = df\_train['issueDate'].apply(lambda x: x-startdate).dt.days

df\_train['earliesCreditLine'] = df\_train['earliesCreditLine'].apply(lambda s: int(s[-4:]))

df\_train['earliesCreditLine']

tags = ['loanAmnt', 'term', 'interestRate', 'installment', 'grade',

'subGrade', 'employmentTitle', 'employmentLength', 'homeOwnership',

'annualIncome', 'verificationStatus', 'issueDateDT', 'earliesCreditLine',

'purpose', 'postCode', 'regionCode', 'dti', 'delinquency\_2years',

'ficoRangeLow', 'ficoRangeHigh', 'openAcc', 'pubRec',

'pubRecBankruptcies', 'revolBal', 'revolUtil', 'totalAcc',

'initialListStatus', 'applicationType', 'title',

'n0', 'n1', 'n2', 'n4', 'n5', 'n6', 'n7', 'n8',

'n9', 'n10', 'n11', 'n12', 'n13', 'n14','yp']

df\_test['grade'] = labelEncoder.fit\_transform(df\_test['grade'])

employmentLength = ['< 1 year','1 year','2 years',

'3 years', '5 years', '4 years',

'6 years', '8 years', '7 years','9 years','10+ years']

j = 0

for i in employmentLength:

df\_test['employmentLength'] = df\_test['employmentLength'].replace(i, j)

j += 1

df\_test = pd.merge(df\_test, \_[["subGrade", "yp"]], on="subGrade", how="left")

df\_test['subGrade'] = labelEncoder.fit\_transform(df\_test['subGrade'])

df\_test['issueDate'] = pd.to\_datetime(df\_test['issueDate'],format='%Y-%m-%d')

startdate = datetime.datetime.strptime('2007-06-01', '%Y-%m-%d')

df\_test['issueDateDT'] = df\_test['issueDate'].apply(lambda x: x-startdate).dt.days

df\_test['earliesCreditLine'] = df\_test['earliesCreditLine'].apply(lambda s: int(s[-4:]))

Standard\_scaler = StandardScaler()

Standard\_scaler.fit(df\_train[tags].values)

x = Standard\_scaler.transform(df\_train[tags].values)

x\_ = Standard\_scaler.transform(df\_test[tags].values)

y = df\_train['isDefault'].values

lgbr = LGBMRegressor(num\_leaves=30

,max\_depth=10

,learning\_rate=0.01

,n\_estimators=13000

,subsample\_for\_bin=5000

,min\_child\_samples=200

,colsample\_bytree=.2

,reg\_alpha=.1

,reg\_lambda=.1

,seed=2020

)

cat = CatBoostRegressor(depth=9,

l2\_leaf\_reg=1,

learning\_rate=0.01,

eval\_metric = 'AUC' ,

border\_count = 128,

bagging\_temperature = 0.9 ,

n\_estimators=16000,

early\_stopping\_rounds=500,

subsample = 0.9,

random\_seed=1,

verbose = 0)

from sklearn.ensemble import VotingRegressor

rg\_model = VotingRegressor([('lgb', lgbr), ('catboost', cat)],n\_jobs=12)

rg\_model.fit(x,y)

pre = pd.DataFrame(rg\_model.predict(x\_),columns=['isDefault'])

results = pd.concat([df\_test['id'],pre],axis = 1)

results.to\_csv('submit.csv', index=False)