Give Me Some Credit

信用卡评分模型

2021年6月3日

壹

数据读取

数据读取

1、分别读取训练集、测试集

```
以下是训练集维数------------------
(150000, 12)
以下是训练集信息------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150000 entries, 0 to 149999
Data columns (total 12 columns):
 #   Column                                Non-Null Count   Dtype  
---  ------                                --------------   -----  
 0   Unnamed: 0                            150000 non-null  int64  
 1   SeriousDlqin2yrs                      150000 non-null  int64  
 2   RevolvingUtilizationOfUnsecuredLines  150000 non-null  float64
 3   age                                   150000 non-null  int64  
 4   NumberOfTime30-59DaysPastDueNotWorse  150000 non-null  int64  
 5   DebtRatio                             150000 non-null  float64
 6   MonthlyIncome                         120269 non-null  float64
 7   NumberOfOpenCreditLinesAndLoans       150000 non-null  int64  
 8   NumberOfTimes90DaysLate               150000 non-null  int64  
 9   NumberRealEstateLoansOrLines          150000 non-null  int64  
 10  NumberOfTime60-89DaysPastDueNotWorse  150000 non-null  int64  
 11  NumberOfDependents                    146076 non-null  float64
dtypes: float64(4), int64(8)
memory usage: 13.7 MB
None
```

```
以下是测试集维数------------------
(101503, 12)
以下是测试集信息------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 101503 entries, 0 to 101502
Data columns (total 12 columns):
 #   Column                                Non-Null Count   Dtype  
---  ------                                --------------   -----  
 0   Unnamed: 0                            101503 non-null  int64  
 1   SeriousDlqin2yrs                      0 non-null       float64
 2   RevolvingUtilizationOfUnsecuredLines  101503 non-null  float64
 3   age                                   101503 non-null  int64  
 4   NumberOfTime30-59DaysPastDueNotWorse  101503 non-null  int64  
 5   DebtRatio                             101503 non-null  float64
 6   MonthlyIncome                         81400 non-null   float64
 7   NumberOfOpenCreditLinesAndLoans       101503 non-null  int64  
 8   NumberOfTimes90DaysLate               101503 non-null  int64  
 9   NumberRealEstateLoansOrLines          101503 non-null  int64  
 10  NumberOfTime60-89DaysPastDueNotWorse  101503 non-null  int64  
 11  NumberOfDependents                    98877 non-null   float64
dtypes: float64(5), int64(7)
memory usage: 9.3 MB
None
```

数据读取

1、分别读取训练集、测试集

S/n	Variable Name	Description	Type
1	**SeriousDlqin2yrs**	**个人经历了逾90天的拖欠或者更糟的情况（区分好坏客户）**	**Y/N**
2	RevolvingUtilizationOfUnsecuredLines	信用卡和个人信贷余额的总余额，减去房地产和没有分期付款的债务（如汽车贷款）除以信用额度总和	percentage
3	age	借款人年龄	integer
4	NumberOfTime30-59DaysPastDueNotWorse	借款人逾期30-59天的次数，但在过去2年没有更差的信用记录	integer
5	DebtRatio	负债比例	percentage
6	MonthlyIncome	月收入	real
7	NumberOfOpenCreditLinesAndLoans	开放贷款的数量和信用额度	integer
8	NumberOfTimes90DaysLate	借款人逾期90天或以上的次数	integer
9	NumberRealEstateLoansOrLines	抵押贷款和房地产贷款的数量	integer
10	NumberOfTime60-89DaysPastDueNotWorse	借款人逾期60-89天的次数，但在过去2年没有更差的信用记录	integer
11	NumberOfDependents	家属人数（配偶，子女等）	integer

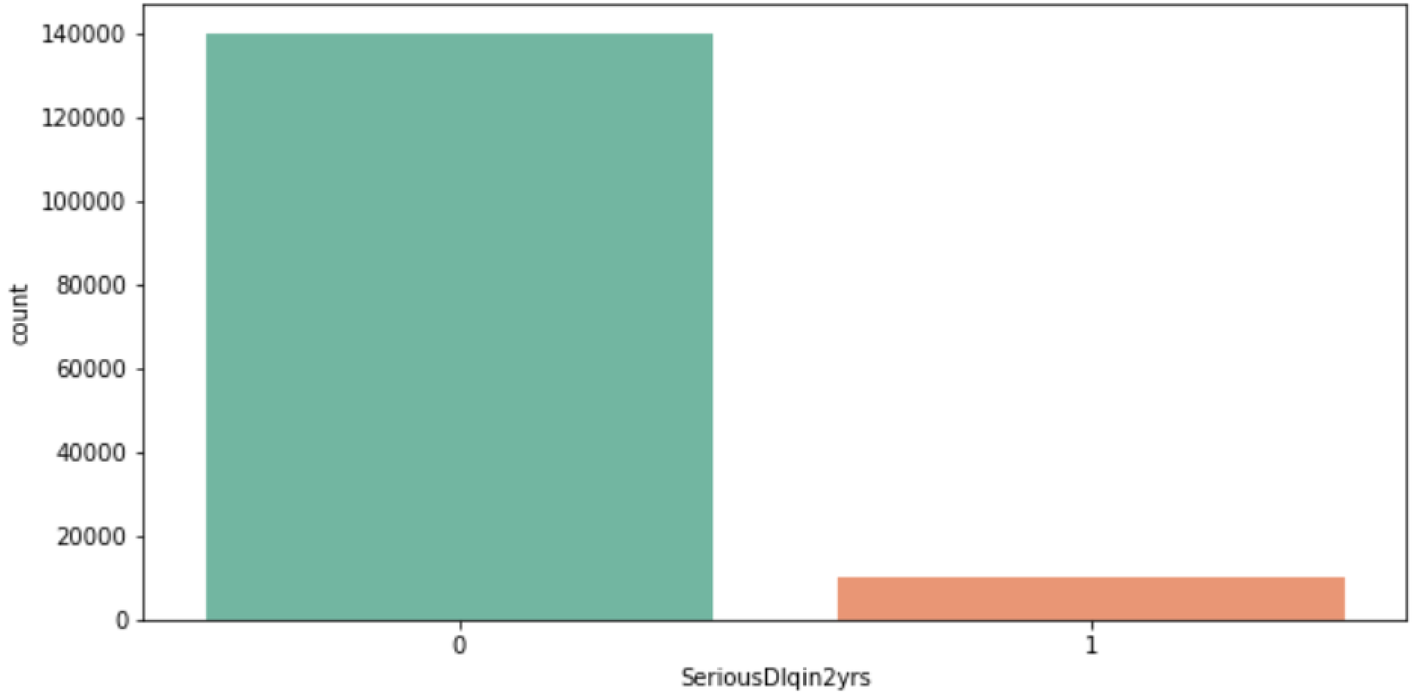
贰

数据分析

数据分析

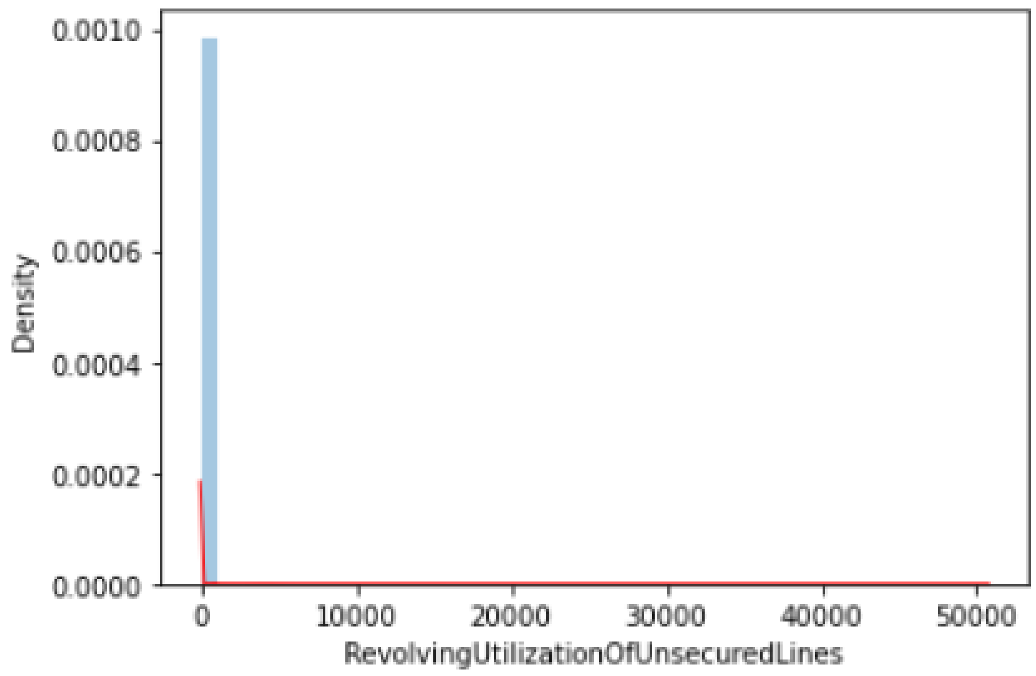
2.1 查看数据特征分布情况

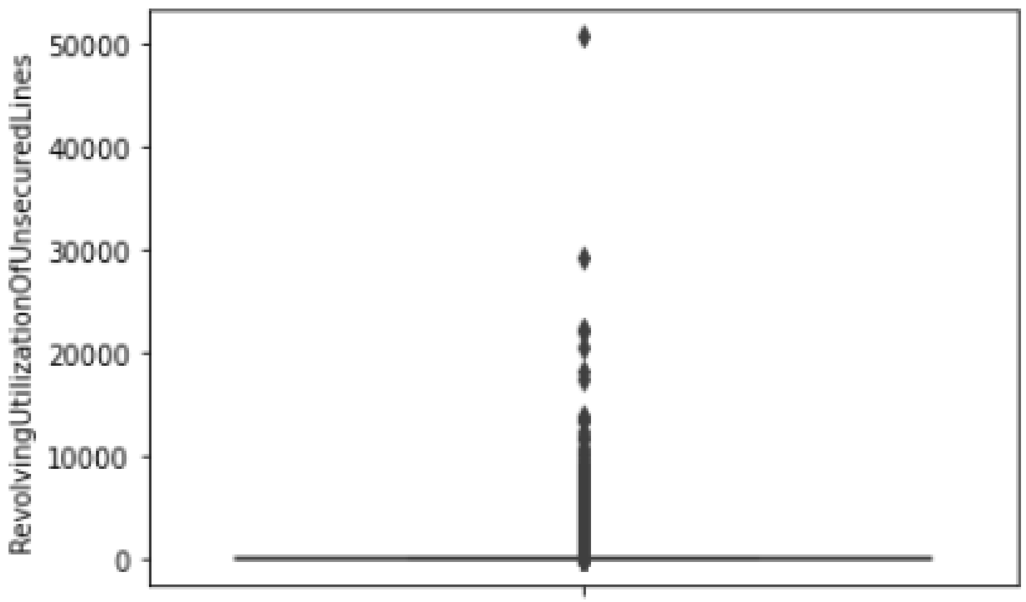
2.1.1 查看好、坏客户分布



数据分析

2.1.2 查看可用额度比值的特征分布



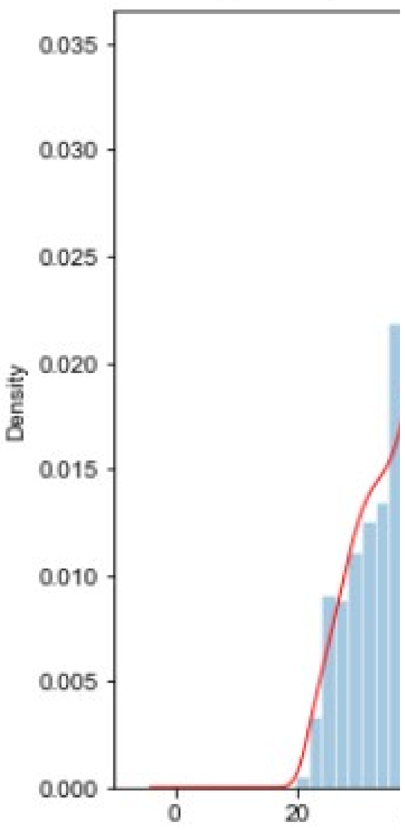


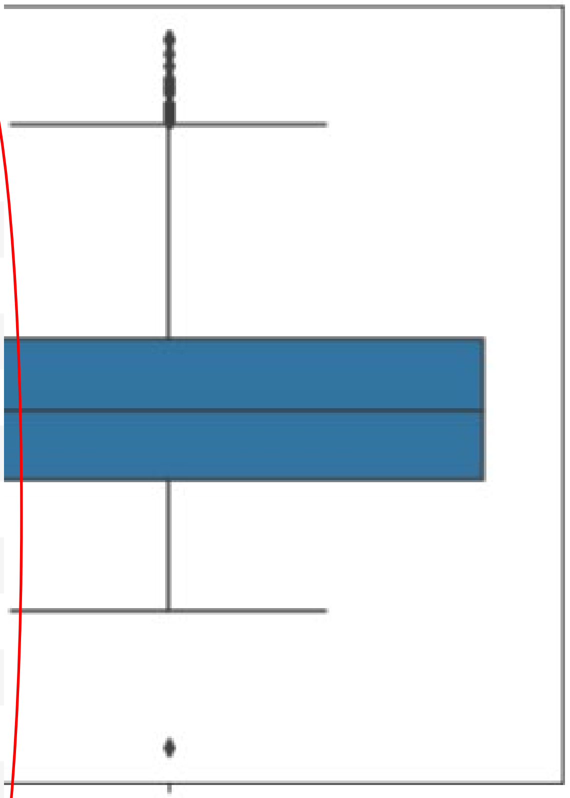
数据分析

2.1.3 查看年龄的

```
#大于100岁的
trainingData[trainingData['age']>100]#较多、连续，可暂时保留
```

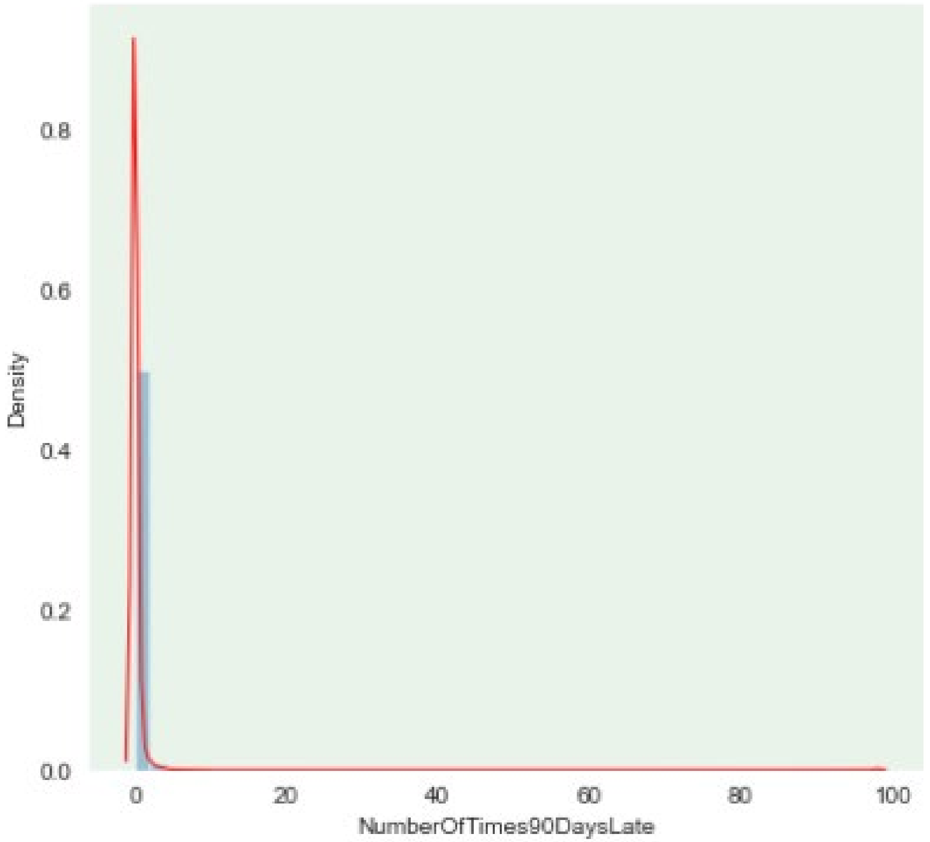
	SeriousDlqin2yrs	RevolvingUtilizationOfUnsecuredLines	age
Unnamed: 0			
7764	0	0.069167	101
19885	0	1.000000	103
25562	0	0.009866	102
40008	0	0.064748	107
56762	0	0.003469	105
57968	0	0.001397	103
90938	0	0.000000	102
93814	0	0.025780	101
96451	0	0.109642	102
105791	0	0.109307	109
116130	1	0.002964	101
135026	0	0.004059	103
138292	0	0.246529	109

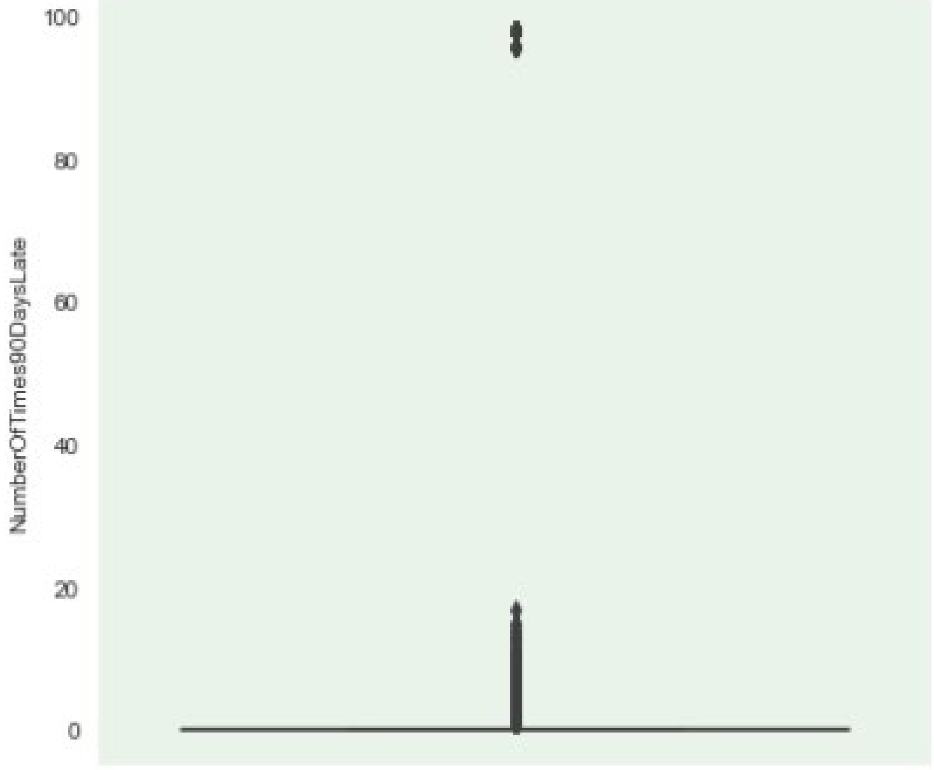




数据分析

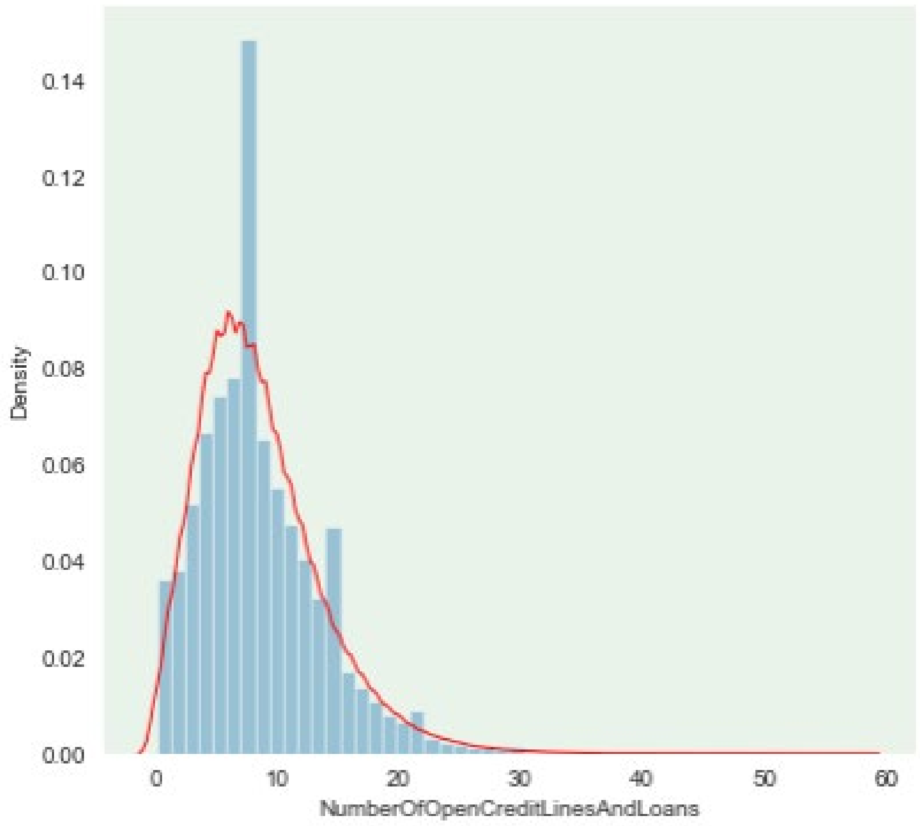
2.1.4 逾期30-59天/60-89天/90天笔数的人数分布

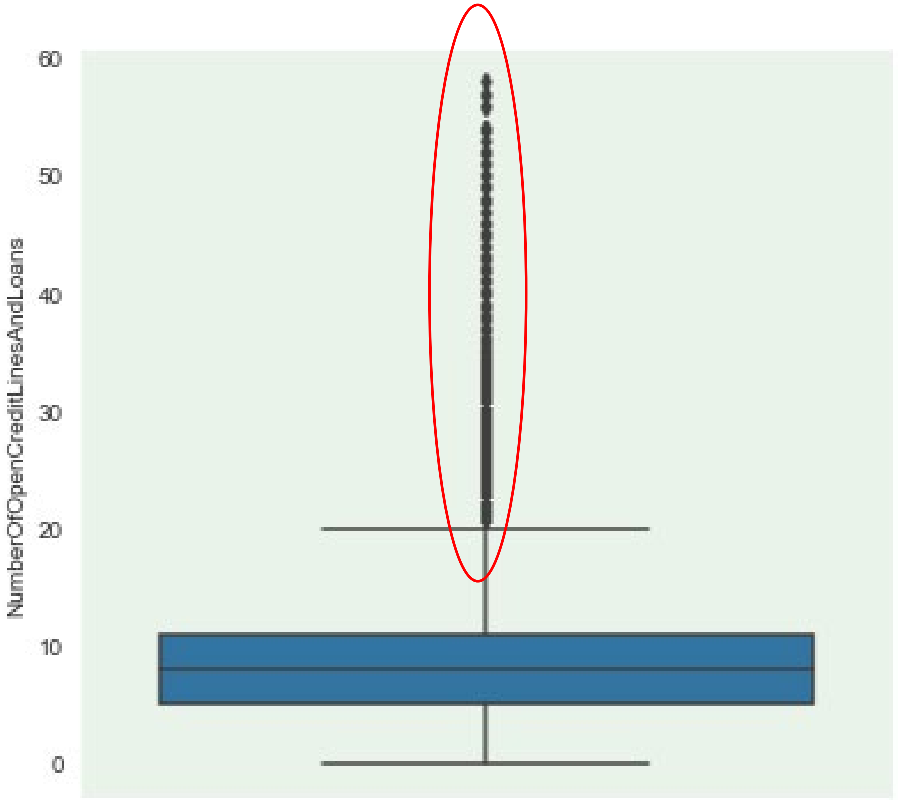




数据分析

2.1.5 查看信贷数量的特征分布





数据分析

2.1.6 家属数量
2.1.7 月收入

```
#查看缺失比例
propotion=(trainingData['SeriousDlqin2yrs'].count()-trainingData['NumberOfDependents'].count())/trainingData['SeriousDlqin2yrs'].count()
print('家属数量缺失比例为%.2f%%'%(propotion*100))
print('结论：缺失比例为2.62%，可直接删除')
```

家属数量缺失比例为2.62%
结论：缺失比例为2.62%，可直接删除

```
#查看缺失比例
propotion=(trainingData['age'].count()-trainingData['MonthlyIncome'].count())/trainingData['age'].count()
print('月收入缺失数量比例为%.2f%%'%(propotion*100))
print('\n结论：由于月收入缺失数量过大，后面采用随机森林的方法填充缺失值')
```

月收入缺失数量比例为19.82%

结论：由于月收入缺失数量过大，后面采用随机森林的方法填充缺失值

叁

数 据 预 处 理

数据预处理

3.1 异常值处理

创建删除异常值函数*myDelete*:

```python
def myDelete(data):
    data=data[data['RevolvingUtilizationOfUnsecuredLines']<1]
    data=data[data['age']>18]
    data=data[data['NumberOfTime30-59DaysPastDueNotWorse']<80]
    data=data[data['NumberOfTime60-89DaysPastDueNotWorse']<80]
    data=data[data['NumberOfTimes90DaysLate']<80]
    data=data[data['NumberOfDependents']<20]
    data=data[data['NumberRealEstateLoansOrLines']<50]
    return data
trainingData=myDelete(trainingData)
testData=myDelete(testData)
```

数据预处理

3.2 缺失值处理

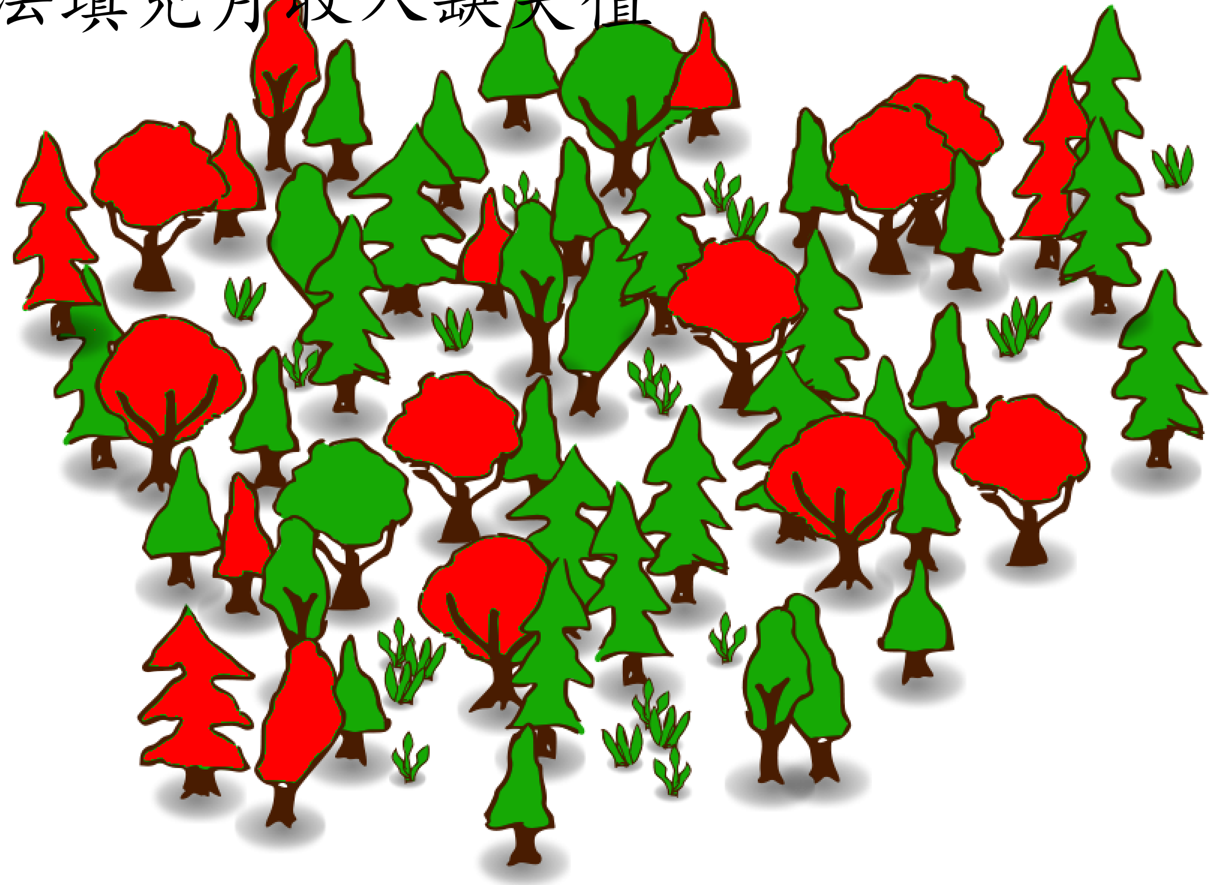
3.2.1 删除家属数量缺失的数据

```
#3.2.1 对家属数量
trainingData=tra                                   nts'].notnull()]
testData=testDat                                   l()]
```

```
(142558, 11)
SeriousDlqin2yrs                              0
RevolvingUtilizationOfUnsecuredLines          0
age                                           0
NumberOfTime30-59DaysPastDueNotWorse          0
DebtRatio                                     0
MonthlyIncome                             25228
NumberOfOpenCreditLinesAndLoans               0
NumberOfTimes90DaysLate                       0
NumberRealEstateLoansOrLines                  0
NumberOfTime60-89DaysPastDueNotWorse          0
NumberOfDependents                            0
dtype: int64
```

数据预处理

3.2.2　随机森林法填充月收入缺失值



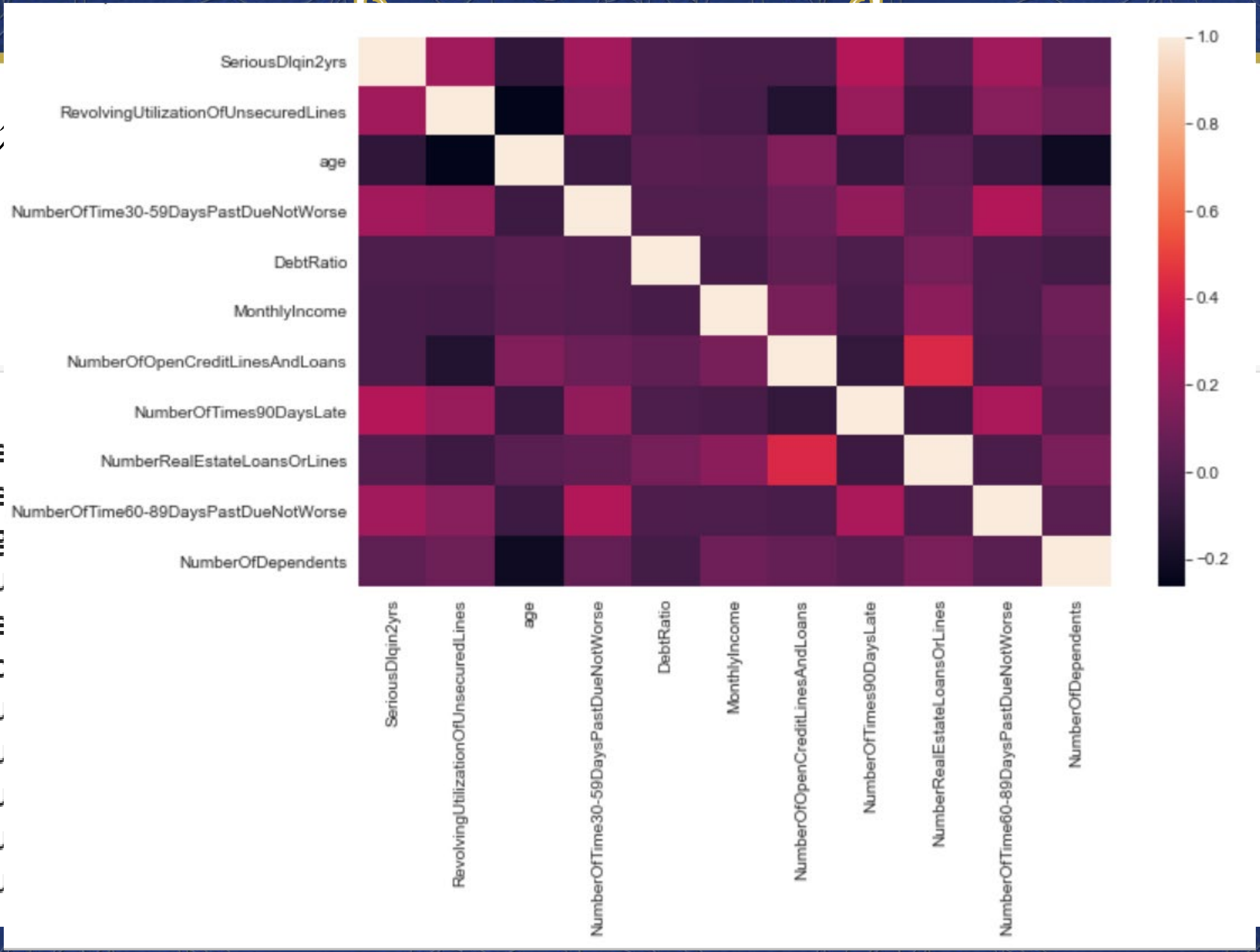
数据预处理

```
In  [41]: #(3.2.2)随机森林法填充月收入缺失值
#创建随机森林填充函数myFiller:
def myFiller(data):
    haveKnown=data[data['MonthlyIncome'].notnull()]
    haveNotKnown=data[data['MonthlyIncome'].isnull()]
    x_0=haveKnown.iloc[:,[1,2,3,4,6,7,8,9,10]]
    y_0=haveKnown.iloc[:,5]
    x_1=haveNotKnown.iloc[:,[1,2,3,4,6,7,8,9,10]]
    randomForest=RandomForestRegressor(random_state=0,n_estimators=200,max_depth=3,n_jobs=-1)
    y_2=randomForest.fit(x_0,y_0).predict(x_1)
    return y_2
```

```
In  [42]: #使用myFiller填充缺失值
#训练集
predictData=myFiller(trainingData)
trainingData.loc[trainingData['MonthlyIncome'].isnull(),'MonthlyIncome']=predictData
print(trainingData.info())
print('\n--------------------------\n')
#测试集
predictData_2=myFiller(testData)
testData.loc[testData['MonthlyIncome'].isnull(),'MonthlyIncome']=predictData_2
print(testData.info())
```

数据预处理

3.3 判断特



Se
Re
ag
Nu
De
Mc
Nu
Nu
Nu
Nu
Nu

肆

特征选择

特征选择

4.0 准备工作——划分数据

- 将训练集数据划分成*training*集和*testing*集，分别用于训练和评估

```
#划分数据
Y=trainingData['SeriousDlqin2yrs']
X=trainingData.iloc[:,1:]
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=0)
training = pd.concat([Y_train, X_train], axis=1)
testing = pd.concat([Y_test, X_test], axis=1)
clasTest = testing.groupby('SeriousDlqin2yrs')['SeriousDlqin2yrs'].count()
```

特征选择

4.1 分箱

- 使用*IV*值进行特征选择并且使用*WOE*(证据权重)对数据进行分箱
- 要对一个变量进行*WOE*编码，需要首先把这个变量进行分组处理

$$WOE_i = \ln(\frac{P(y_i)}{P(y_n)}) = \ln(\frac{y_i / y_T}{n_i / n_T})$$

- *WOE*表示的实际上是“当前分组中响应客户占所有响应客户的比例”和“当前分组中没有响应的客户占所有没有响应的客户的比例”的差异。

特征选择

4.1 分箱

- IV值（信息价值），用来衡量自变量的预测能力。

$$IV_i = \left(P(y_i) - P(n_i)\right) \times WOE_i = \left(\frac{y_i}{y_T} - \frac{n_i}{n_T}\right) \times \ln\left(\frac{y_i/y_T}{n_i/n_T}\right)$$

- 有了一个变量各分组的IV值，把各分组的IV相加就可得整个变量的IV值。

$$IV = \sum_{i=1}^{n} IV_i$$

特征选择

4.1 分箱

4.1.1 连续性变量——最优分箱

```
def autoBin(target,data,n=10): #data为待分箱变量，n为分箱数量
```

月收入：

```
Bucket
(-0.001, 3416.0]       0.068875
(3416.0, 6900.0]       0.064987
(6900.0, 1794060.0]    0.046244
Name: rate, dtype: float64
---------------------
分箱结果:
      min        max   bad  total      rate       woe   badattr  goodattr
0     0.0     3416.0  2620  38040  0.068875  0.146609  0.382593  0.330420
1  3417.0     6900.0  2473  38054  0.064987  0.084332  0.361127  0.331922
2  6902.0  1794060.0  1755  37951  0.046244 -0.275768  0.256279  0.337659
IV值为:
0.03255400450513984
```

特征选择

4.1 分箱

4.1.2 离散型变量——手动分箱

逾期30-59天

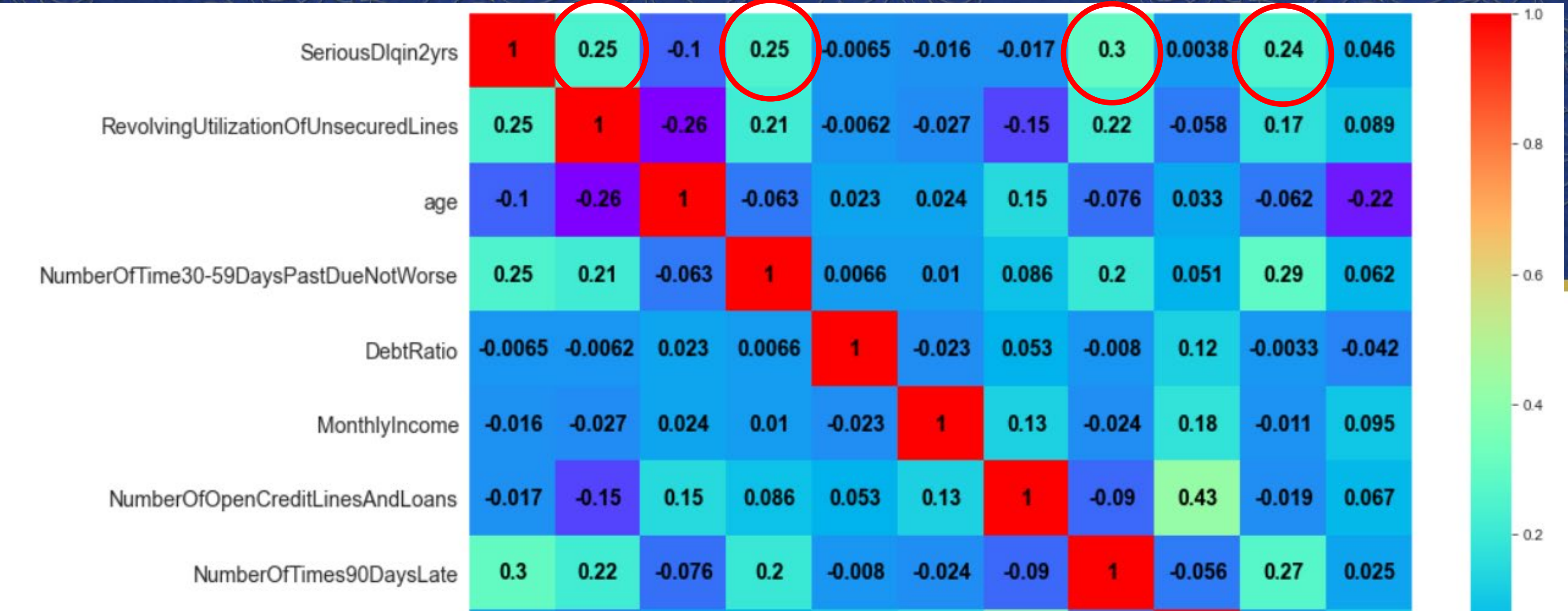
```python
#手动分箱法
def myBin(target,data,cut):
```

```
(5.0, inf]      0.496732
Name: 
-------
分箱结
   min                 goodattr
0    0                 0.868019
1    1                 0.096523
2    2                 0.030663
3    4                 0.004077
4    6                 0.000718
IV值为:
0.6356
```

```python
ninf = float('-inf')#负无穷大
pinf = float('inf')#正无穷大
cutx3 = [ninf, 0, 1, 3, 5, pinf]
cutx6 = [ninf, 1, 2, 3, 5, pinf]
cutx7 = [ninf, 0, 1, 3, 5, pinf]
cutx8 = [ninf, 0,1,2, 3, pinf]
cutx9 = [ninf, 0, 1, 3, pinf]
cutx10 = [ninf, 0, 1, 2, 3, 5, pinf]
```

4.2 特

4.2.1




```python
corr=training.corr()#计算相关性系数
x=list(corr.index)
y=list(corr.index)
fig=plt.figure(figsize=(14,10))
ax1=fig.add_subplot(1,1,1)
#绘制相关性系数热力图
sns.heatmap(corr,annot=True,ax=ax1,cmap='rainbow',annot_kws={'size':14,'weight':'bold','color':'black'})
ax1.set_xticklabels(x,rotation=90,fontsize=14)
ax1.set_yticklabels(y,rotation=0,fontsize=14)
plt.show()
```

4.2 特征选择

4.2.2 *IV*值筛选

- 4.2.1中的相关性分 VI

（证据权重）作为变量

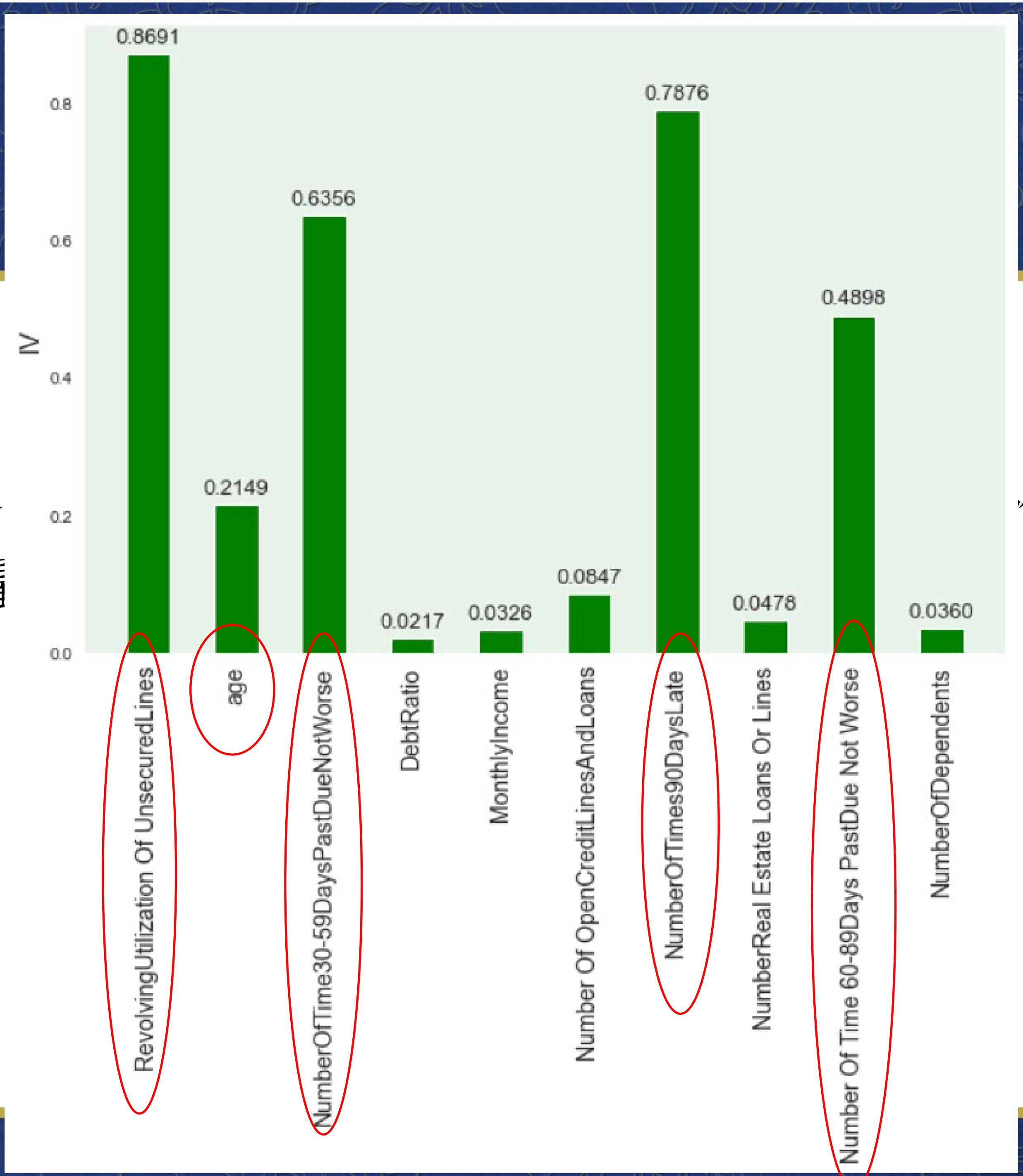
- 评判标准：

<0.02:*unpredictive*

0.02–0.1:*weak*

0.1–0.3:*medium*

>0.3:*strong*



五

模型建立

模型建立

5.1 一些准备

5.1.1 将筛选后的变量根据分箱结果转换为WOE值

	RevolvingUtilizationOfUnsecuredLineswoe	agewoe	NumberOfTime30-59DaysPastDueNotWorsewoe	NumberOfTimes90DaysLatewoe	NumberOfTime60-89DaysPastDueNotWorsewoe
Unnamed: 0					
13017	1.047	-0.207	-0.472	-0.347	-0.242
60720	-1.138	-0.986	-0.472	-0.347	-0.242
67395	1.047	0.130	0.883	-0.347	-0.242
56109	-1.197	-0.986	-0.472	-0.347	-0.242
130935	-1.197	-0.504	-0.472	-0.347	-0.242

模型建立

5.1 一些准备

5.1.2 构建自变量和因变量，剔除对因变量影响不明显的变量

X =自变量

Y =因变量

```
Y=training['SeriousDlqin2yrs']    #因变量
#剔除对因变量影响不明显的变量
X=training.drop(['SeriousDlqin2yrs','DebtRatio','MonthlyIncome',
                 'NumberOfOpenCreditLinesAndLoans','NumberRealEstateLoansOrLines',
                 'NumberOfDependents'],axis=1)
X=training.iloc[:,-5:]
X.head(5)
```

模型建立

```
Optimization terminated successfully.
         Current function value: 0.176636
         Iterations 8
<statsmodels.discrete.discrete_model.BinaryResultsWrapper object at 0x0000023A091EB880>
                           Logit Regression Results
==============================================================================
Dep. Variable:      SeriousDlqin2yrs   No. Observations:               114045
Model:                         Logit   Df Residuals:                   114039
Method:                          MLE   Df Model:                            5
Date:               Sun, 30 May 2021   Pseudo R-squ.:                  0.2222
Time:                       10:29:45   Log-Likelihood:                -20144.
converged:                      True   LL-Null:                       -25899.
Covariance Type:           nonrobust   LLR p-value:                     0.000
========================================================================================================
                                            coef    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------------------------------
const                                    -2.7251      0.015   -183.611      0.000      -2.754      -2.696
RevolvingUtilizationOfUnsecuredLineswoe   0.6565      0.016     41.094      0.000       0.625       0.688
agewoe                                    0.5041      0.033     15.082      0.000       0.439       0.570
NumberOfTime30-59DaysPastDueNotWorsewoe   0.5567      0.016     34.615      0.000       0.525       0.588
NumberOfTimes90DaysLatewoe                0.5965      0.013     44.353      0.000       0.570       0.623
NumberOfTime60-89DaysPastDueNotWorsewoe   0.4276      0.018     24.052      0.000       0.393       0.462
========================================================================================================
```

六

模型评估

模型评估

6.1 *AUC*评估

- 使用建模初期保留的*testing*集，利用*sklearn.metrics*，比较两个分类器，自动计算*ROC*和*AUC*

```
#评估
X3=sm.add_constant(test_X)
resu=result.predict(X3)
print(resu)
fpr,tpr,threshold=metrics.roc_curve(test_Y,resu)   #评估算法
rocauc=metrics.auc(fpr,tpr)   #计算AUC
#绘图
plt.figure(figsize=(10,8))  #只能在这里面设置
plt.plot(fpr,tpr,'b',label='AUC=%0.2f'% rocauc)
plt.legend(loc='lower right',fontsize=14)
plt.plot([0, 1], [0, 1], 'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.ylabel('TPR',fontsize=16)
plt.xlabel('FPR',fontsize=16)
plt.show()
```

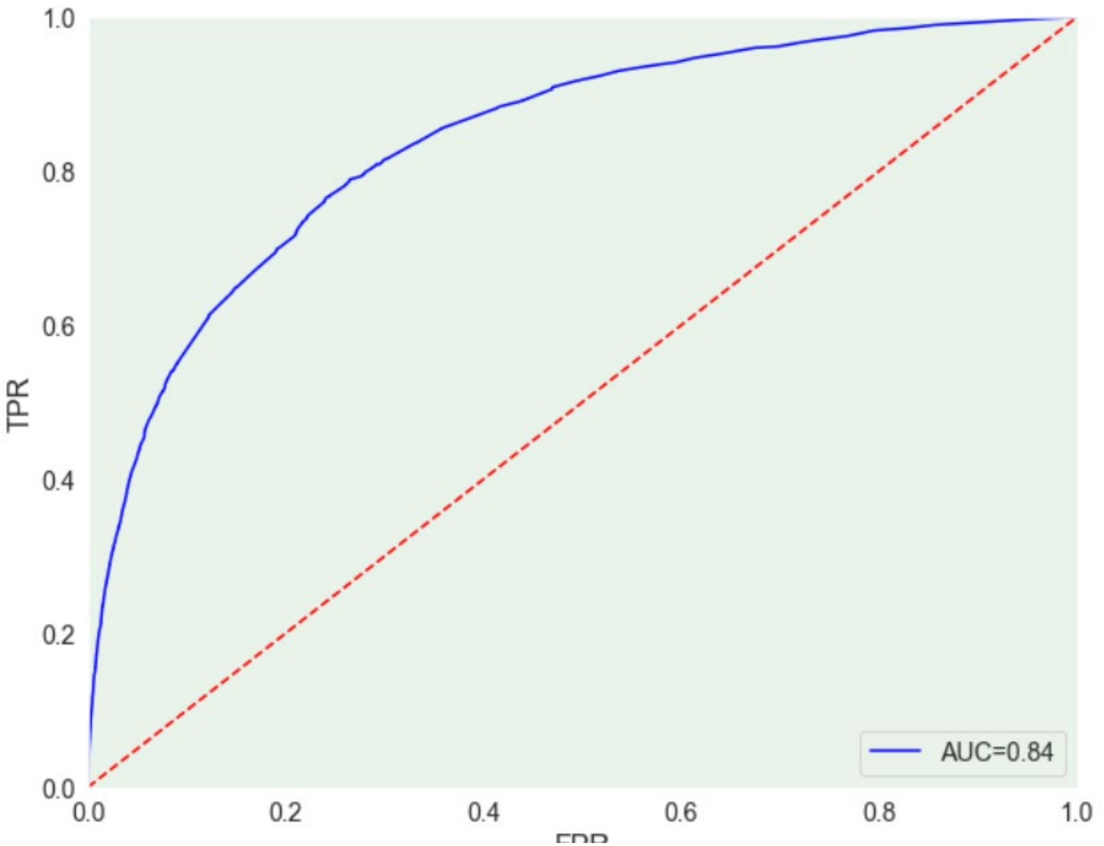
模型评估

6.1 *AUC*评估

TPR： 真阳性（正判）率，越高说明模型精确度越高

FPR： 假阳性（错判）率，越高说明模型敏感度越高

故而*ROC*曲线向上拱的幅度越大，说明拟合效果越好，但仍需警惕过拟合



可见*AUC*为0.84，说明这个模型的拟合效果不错，正确率较高。

模型评估

6.2 *KS*指标评估

• 引入*KS*值来评估本模型。这个数值越大，说明模型将好坏客户区分的能力越强。

判别标准如下：

KS: $< 20\%$：差

KS: $20\% - 40\%$：一般

KS: $41\% - 50\%$：好

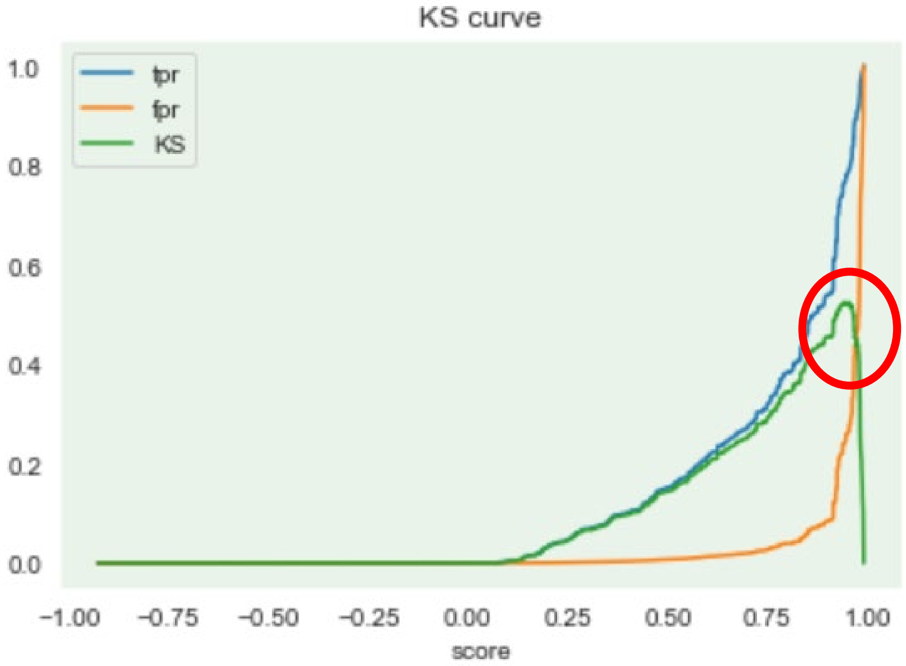
KS: $51\% - 75\%$：非常好

KS: $> 75\%$：过高，需要谨慎的验证模型

*KS*曲线的计算方式为$KS = TPR - FPR$，*KS*值是*KS*曲线上的最大值。故通常来说越大越好。但过大的*KS*值意味着模型存在过拟合风险

这里*KS*值约为0.524，说明其区分效果不错

```
In [56]: #KS指标：用以评估模型对
#计算累计坏客户与累计好
fig,ax = plt.subplots()
ax.plot(1-threshold,tpr
ax.plot(1-threshold,fpr
ax.plot(1-threshold,tpr-
plt.xlabel('score')
plt.title('KS curve')
plt.xlim=([0.0,1.0])
plt.ylim=([0.0,1.0])
plt.figure(figsize=(20,
legend=ax.legend(loc='u
plt.show()
max(tpr-fpr)
```




```
<Figure size 1440x1440 with 0 Axes>
0.5241713995900088
```

七

创建信用评分卡

创建信用评分卡

7.1 评分卡建立

- $Score = \sum_1^n \left(WOE_i \times coef_i + \frac{coef_0}{n} \right) \times factor + offset$
- $factor = p/\log(2)$ p为好坏比翻一倍时评分增加的分数
- $offset = b - p \times \frac{\log(o)}{\log(2)}$ b为基础分值，o为基础分值对应的好坏比

In [58]:
```
#6、创建信用评分卡
#6.1 在建立标准评分卡之前，还需要设定几个评分卡参数：基础分值、 PDO（比率翻倍的分值）和好坏比。
#我们取600分为基础分值b，取20为PDO （每高20分好坏比翻一倍），好坏比O取20。
p=20/np.log(2) #比例因子
q=600-20*np.log(20)/np.log(2) #偏移量
x_coe=[-2.7251,0.6565,0.5041,0.5576,0.5965,0.4276] #上面计算得出的回归系数
base=round(q+p*x_coe[0],0) #基础得分
#总分=基础分+各部分得分
def Score(coe,woe,factor):
    scores=[]
    for k in woe:
        score=round(coe*k*factor,0)
        scores.append(score)
    return scores
#每一项得分
x1_score=Score(x_coe[1],x1_woe,p) #注：'RevolvingUtilizationOfUnsecuredLines'
x2_score=Score(x_coe[2],x2_woe,p) #注：'age'
x3_score=Score(x_coe[3],woex3,p) #注：'NumberOfTime30-59DaysPastDueNotWorse'
x7_score=Score(x_coe[4],woex7,p) #注：'NumberOfTimes90DaysLate'
x9_score=Score(x_coe[5],woex9,p) #注：'NumberOfTime60-89DaysPastDueNotWorse'
```

7.2 计算得分

- 根据评分卡的计分规则算出来各个数据的得分

In [60]:
```
#计算得分
trainingData['BaseScore']=np.zeros(len(trainingData))+base
trainingData['x1']=compute(trainingData['RevolvingUtilizationOfUnsecuredLines'],x1_cut,x1_score)
trainingData['x2']=compute(trainingData['age'], x2_cut, x2_score)
trainingData['x3']=compute(trainingData['NumberOfTime30-59DaysPastDueNotWorse'], cutx3, x3_score)
trainingData['x7']=compute(trainingData['NumberOfTimes90DaysLate'], cutx7, x7_score)
trainingData['x9']=compute(trainingData['NumberOfTime60-89DaysPastDueNotWorse'],cutx9,x9_score)
trainingData['Score']=trainingData['x1']+trainingData['x2']+trainingData['x3']+trainingData['x7']+trainingData['x9']+base
#选取需要的列，就是评分列
scoreTable1=trainingData.iloc[:,[0,-7,-6,-5,-4,-3,-2,-1]]
scoreTable1.head(5)
```

创建信用评分卡

训练集效果展示

	A	B	C	D	E	F	G	H	I
1	nnamed:	eriousDlqin2yr	BaseScore	ilizationOfUns	age	e30-59DaysPast	erOfTimes90Day	e60-89DaysPast	Score
2	1	1	435	20	3	28	34	23	543
3	2	0	435	20	4	14	34	23	530
4	3	0	435	20	5	28	47	23	558
5	4	0	435	-5	7	14	34	23	508
6	5	0	435	20	3	28	34	23	543
7	6	0	435	-5	-14	14	34	23	487
8	7	0	435	-5	-3	14	34	23	498
9	8	0	435	20	5	14	34	23	531
10	10	0	435	-5	-3	14	34	23	498
11	11	0	435	20	7	14	34	23	533
12	12	0	435	-23	2	14	34	23	485
13	13	0	435	-23	3	14	34	23	486
14	14	1	435	20	4	38	58	33	588
15	15	0	435	-23	-14	14	34	23	469
16	16	0	435	20	-12	14	34	23	514

八

对测试集进行预测转化为信用评分卡

对 测 试 集 进 行 预 测
转 化 为 信 用 评 分 卡

8.1 处理测试集

In [62]:

```
#7、利用模型预测测试集
#7.1 测试集转化为WOE值
testData=toWOE(testData,x1_name,x1_woe,x1_cut)
testData=toWOE(testData,x2_name,x2_woe,x2_cut)
testData=toWOE(testData,x3_name,woex3,cutx3)
testData=toWOE(testData,x7_name,woex7,cutx7)
testData=toWOE(testData,x9_name,woex9,cutx9)
#自变量，剔除对因变量影响不明显的变量
testData=testData.drop(['DebtRatio','MonthlyIncome', 'NumberOf(
#测试集的特征和标签
test_X=testData.iloc[:,-5:]
test_Y=testData.iloc[:,0]
#7.2 评估
X_=sm.add_constant(test_X)
list=result.predict(X_)
testData['predict']=list
```

对测试集进行预测
转化为信用评分卡

In [63]:

```python
#7.3 对测试集进行评分
testData['BaseScore']=np.zeros(len(testData))+base
testData['x1'] = compute(testData['RevolvingUtilizationOfUnsecuredLines'], x1_cut, x1_score)
testData['x2'] = compute(testData['age'], x2_cut, x2_score)
testData['x3'] = compute(testData['NumberOfTime30-59DaysPastDueNotWorse'], cutx3, x3_score)
testData['x7'] = compute(testData['NumberOfTimes90DaysLate'], cutx7, x7_score)
testData['x9'] = compute(testData['NumberOfTime60-89DaysPastDueNotWorse'],cutx9,x9_score)
testData['Score'] = testData['x1'] + testData['x2'] + testData['x3'] + testData['x7'] +testData['x9']  + base
#选取需要的列，就是评分列
scoretable2=testData.iloc[:,[0,-8,-7,-6,-5,-4,-3,-2,-1]]
print(scoretable2.head(5))
```

```
            SeriousDlqin2yrs   predict  BaseScore    x1   x2    x3    x7  \
Unnamed: 0
1                        NaN  0.076739      435.0  20.0  4.0  14.0  34.0
2                        NaN  0.027417      435.0  -5.0 -3.0  14.0  34.0
3                        NaN  0.015523      435.0 -22.0 -7.0  14.0  34.0
4                        NaN  0.073192      435.0  -5.0  5.0  28.0  34.0
5                        NaN  0.086396      435.0  20.0  7.0  14.0  34.0

              x9  Score
Unnamed: 0
1           23.0  530.0
2           23.0  498.0
3           23.0  477.0
4           23.0  520.0
5           23.0  533.0
```

对测试集进行预测
转化为信用评分卡

8.2 成

In [64]:

otWorse',

predict	BaseScore	ilizationOfUns	age	e30-59DaysPast	rOfTimes90Days	e60-89DaysPast	Score
0.076739446	435	20	4	14	34	23	530
0.027417396	435	-5	-3	14	34	23	498
0.015523241	435	-22	-7	14	34	23	477
0.073192089	435	-5	5	28	34	23	520
0.08639633	435	20	7	14	34	23	533
0.023695521	435	-5	-7	14	34	23	494
0.076383066	435	20	2	14	34	23	528
0.022138259	435	-22	-14	28	34	23	484
0.010910252	435	-23	-12	14	34	23	471
0.019593406	435	-23	5	14	34	23	488
0.018344733	435	-22	2	14	34	23	486
0.010910252	435	-23	-12	14	34	23	471
0.03872072	435	-5	7	14	34	23	508
0.06207429	435	20	-3	14	34	23	523
0.022034074	435	-22	7	14	34	23	491
0.042777829	435	20	-14	14	34	23	512
0.020351458	435	-22	5	14	34	23	489
0.186177157	435	20	7	14	34	33	543
0.076739446	435	20	4	14	34	23	530
0.019322574	435	-22	3	14	34	23	487
0.023695521	435	-5	-7	14	34	23	494
0.010910252	435	-23	-12	14	34	23	471
0.615888459	435	20	-12	28	47	33	551
0.069681822	435	-5	2	28	34	23	517
0.018344733	435	-22	2	14	34	23	486
0.076739446	435	20	3	14	34	23	529

一点感想

加以改进？

Thanks for your listening!