第11章　信用评分模型实例

Jupyter金融应用从入门到实践

场景介绍

信用评分是指信用评估机构利用信用评分模型对贷款客户信用信息进行量化分析，以分值形式表述的信用评分来给客户授信或发放贷款可以达到风险分类、简化审批流程、降低贷款周期和加快放款速度等效果。信用评分主要是根据客户过去一段时间的交易信息、还款状况，以及一些身份属性而产生的信用评分模型，以作为之后贷款批准与否的参考，也可以用来预测客户发生还款逾期的风险程度。信用评分可以看作一个客户的二分类问题，也可以看作对客户信用程度或风险程度的预测，前者的取值为0和1，后者则是0到1之间的连续取值。在本章中，我们将基于kaggle的贷款数据集（GiveMeSomeCredit）来进行案例讲解。

1. 缺失值

通过以下示例，你可以看到变量 MonthlyIncome 和 NumberOfDependents 都有数据缺失。

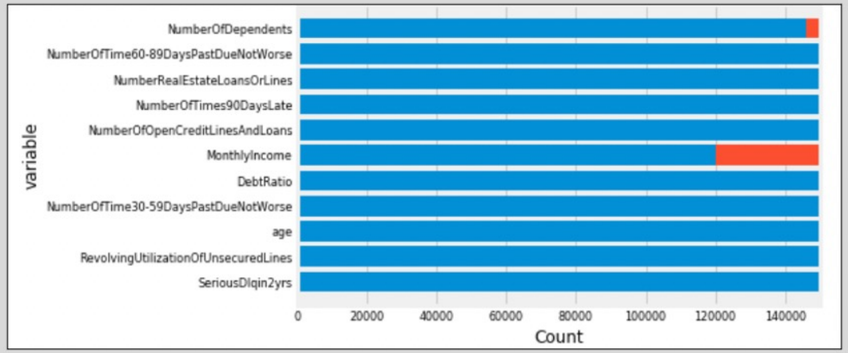
[In]
```
1.  # 统计缺失值
2.  data.isnull().sum(axis=0)
```

[Out]
```
SeriousDlqin2yrs                        0
RevolvingUtilizationOfUnsecuredLines    0
age                                     0
NumberOfTime30-59DaysPastDueNotWorse    0
DebtRatio                               0
MonthlyIncome                       29731
NumberOfOpenCreditLinesAndLoans         0
NumberOfTimes90DaysLate                 0
NumberRealEstateLoansOrLines            0
NumberOfTime60-89DaysPastDueNotWorse    0
NumberOfDependents                   3924
dtype: int64
```

经过可视化之后，这一结果更为直观。

[In]
```
1.  # 可视化缺失值
2.  cols = data.columns
3.  nullcounts = []
4.  value_counts = []
5.  for col in cols:
6.      nullcounts.append(data[col].isnull().sum())
7.      value_counts.append(data[col].shape[0] - data[col].isnull().sum())
8.
9.  fig, ax = plt.subplots()
10. ax.barh(cols, value_counts, label='not missing')
11. ax.barh(cols, nullcounts, label='missing', left=value_counts)
12. ax.set_xlabel('Count')
13. ax.set_ylabel('variable')
14. plt.show()
```

[Out]



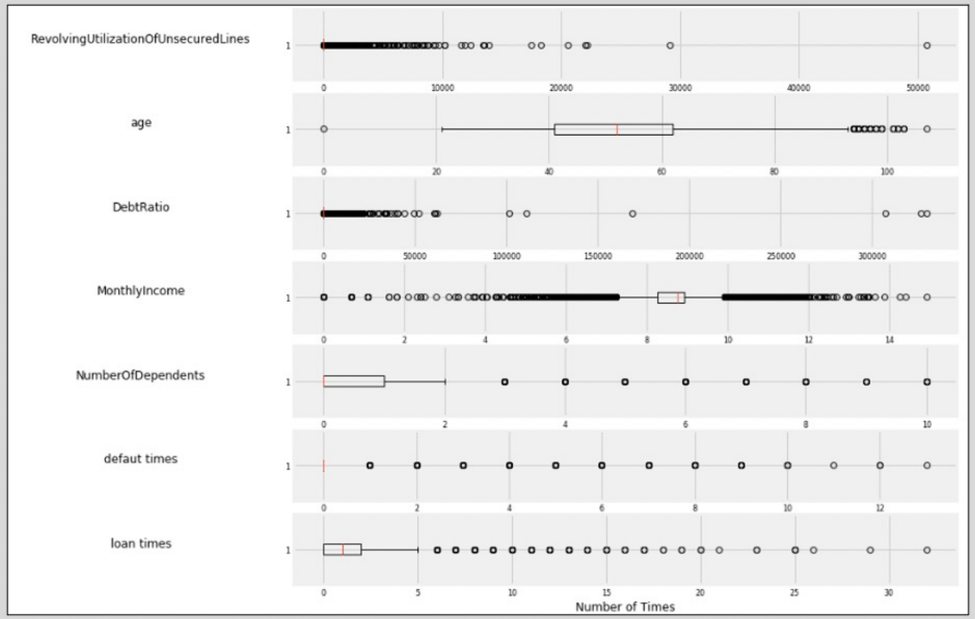
2．异常值

异常值通常是极端值，指偏离大多数抽样数据的数值。可以通过箱线图观察所有变量的取值分布发现异常值。

[In]

```
1.  # 批量画箱线图
2.  x_cols =data.columns.tolist().copy()
3.  x_cols.remove('SeriousDlqin2yrs')
4.  fig, axes = plt.subplots(len(x_cols),1, figsize=(11,11))
5.  i = 0
6.  for c in x_cols:
7.      #ax=plt.boxplot(train_data[c], vert=False,ax = axes[i])
8.
9.      ax = axes[i]
10.     ax.boxplot(data[c],vert=False)
11.     ax.set_ylabel(c, rotation=0,labelpad=150)
12.     ax.set_xlabel("Number of Times")
13.     i +=1
14. plt.show()
```

[Out]



3．数据预处理

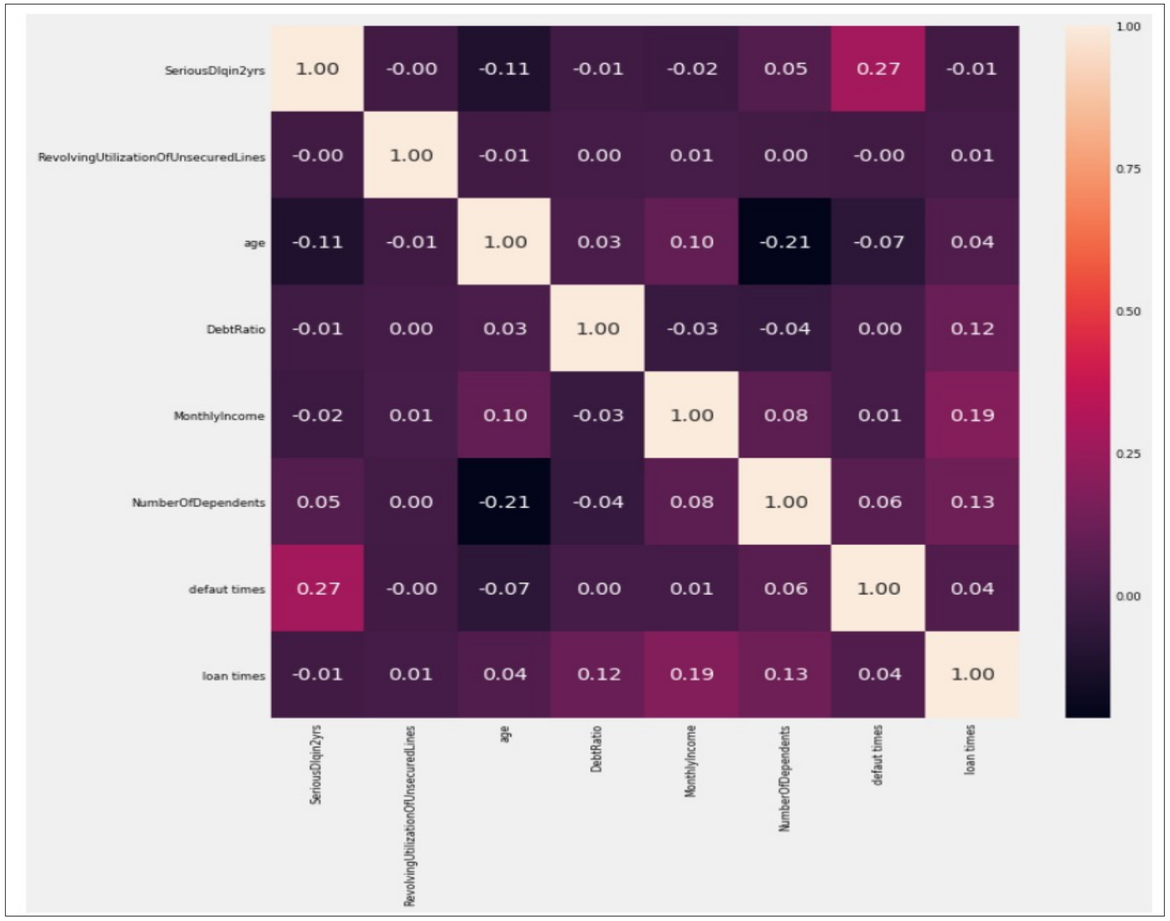
缺失值和异常值可以按行处理或按列删除、填充（插值）和特征转换。其中删除会损失数据信息，需谨慎操作，大多时候都针对较少的数据执行删除操作。

本例采用以下方式处理缺失值和异常值。

- NumberOfDependents，删除缺失值，超过10以上都取为10。
- age，用各类别中的均值填充缺失值和零值异常值。
- NumberOfTime30-59DaysPastDueNotWorse 、 NumberOfTimes90DaysLate 、 NumberOf- Time60-89DaysPastDueNotWorse等，删除异常值，并通过求和转换为新的变量defaut times，代表跨不同期限的违约次数。
- MonthlyIncome，用均值填充缺失值后取对数，压缩取值范围。
- NumberRealEstateLoansOrLines，删除异常值后与NumberOfOpenCreditLinesAndLoans合并成新的变量loan times，代表不同类型贷款总的贷款次数。

1．相关性

在图11-1中，各解释变量之间不存在高度相关。值得注意的是被解释变量SeriousDlqin2yrs与各解释变量之间的相关性中，RevolvingUtilizationOfUnsecuredLines几乎与解释变量不相关



2．信息值

信息值用于检查各变量的预测强度。在计算信息值（IV）之前，我们先来了解变量分箱和证据权重（WOE）。变量分箱（binning）就是将连续数据分组，使连续变量离散化。这是由于在信用评分中大多数解释变量都是具有多个取值的整型数据或者本身就是连续类型，要对一个变量进行WOE和IV计算，需要首先把这个变量进行分组处理。常用的分箱方式包括等距分段、等深分段或最优分箱。在这里使用决策树获得最优分箱的边界值列表。

证据权重（Weight Of Evidence，WOE）代表该组别客户的区别程度，在分箱的基础上就可以计算 WOE。

$$\text{WOE} = \ln\left(\frac{\text{正样本比例}}{\text{负样本比例}}\right)$$

WOE 的绝对值越高，正负样本的区分度也就越高。IV 具有同样的效果。

$$\text{IV} = \Sigma(\text{正样本比例} - \text{负样本比例})\,\text{WOE}$$

即：

$$\text{IV} = \Sigma(\text{正样本比例} - \text{负样本比例})\ln\left(\frac{\text{正样本比例}}{\text{负样本比例}}\right)$$

信息值（IV）	解释能力
[0,0.03)	无预测能力
[0.03,0.10)	低
[0.10,0.30)	中
[0.30,0.50)	高
[0.50,+∞)	极高

数据采样

数据采样主要解决分类数据的类别不平衡性，采样方式主要有上采样和下采样。上采样是重复采样少数类直到两类数据趋于平衡，下采样是不断剔除多数类直到两类数据趋于平衡。上采样消耗额外的内存但保留了样本有效信息，下采样节省内存但损失了精度。

1．数据平衡性

SeriousDlqin2yrs 为被解释变量，代表客户是否发生 90 天及以上的逾期行为，发生则为 1，未发生则为 0。其中正样本 136125 条，负样本 9712 条。负样本占比约为 6.6%，属于严重有偏数据。

```
[In]   print(data[y_col].value_counts())    # 打印被解释变量各类数量
```

```
[Out]  0    136125
       1      9712
       Name: SeriousDlqin2yrs, dtype: int64
```

2．上采样

为不损失训练有效性，采用上采样增强数据，采样后正负样本比例接近 2∶1。

[In]

```
#上采样
from sklearn.model_selection import train_test_split
# 切分数据
data=data[x_col+[y_col]]
X_train, X_test, y_train, y_test  = train_test_split(data[x_col], data[y_col],
                                    test_size=0.3,random_state=2020)

train_data = pd.concat([X_train, y_train], axis=1)  # 合并 x、y 为一个 DataFrame，方便后续计算
# bdf.columns = ['x', 'y']

# 采样规模，只补一半
sample_scale=int(0.5*(train_data.iloc[:,-1].value_counts().max()-train_data.iloc[:,-1].value_counts().min()))
# 采样
upsampling=train_data[train_data.iloc[:,-1]==1].sample(n=sample_scale,replace=True,axis=0,random_state=2020)

# 聚合采样和原数据
sample=pd.concat([upsampling,train_data])
# 切分数据
X_train, y_train=sample.iloc[:,:-1],sample.iloc[:,-1]

sample[y_col].value_counts()
```

[Out]

```
0    95309
1    51042
Name: SeriousDlqin2yrs, dtype: int64
```

建立模型

本案例应用支持向量机、逻辑回归和随机森林这3个分类模型。对于信用评分，可以预测0还是1的分类标签，也可以预测贷款违约的概率。在这里，只选取前者作为模型预测目标。

[In]

```
# 导入模型相关的库
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression

# 创建字典用于保存模型
models={}
for i,model in enumerate([SVC(),LogisticRegression(),RandomForestClassifier()]):
    model.fit(X_train, y_train) # 训练
    models[i]=model
    print('model',model) # 打印模型参数
    print('-'*60)
```

[Out]

```
model SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
------------------------------------------------------------
model LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
------------------------------------------------------------
model RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
------------------------------------------------------------
```

验证评估

[In]

```
# 对比随机森林在训练集和测试集上的表现
y_train_hat=models[2].predict(X_train)
y_test_hat=models[2].predict(X_test)
print('训练集上：')
print('分类报告\n',classification_report(y_train, y_train_hat))

print('\n测试集上：')
print('分类报告\n',classification_report(y_test, y_test_hat))
```

[Out]

训练集上：
分类报告

	precision	recall	f1-score	support
0	1.00	1.00	1.00	95309
1	1.00	1.00	1.00	51042
avg / total	1.00	1.00	1.00	146351

测试集上：
分类报告

	precision	recall	f1-score	support
0	0.94	0.98	0.96	40816
1	0.34	0.15	0.21	2936
avg / total	0.90	0.92	0.91	43752

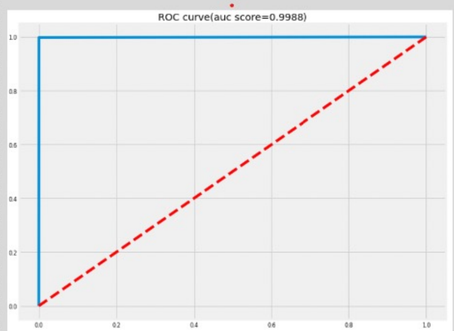
[In]

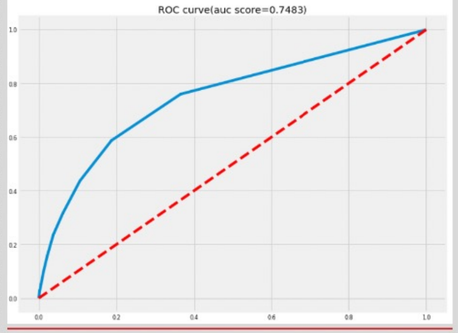
```
from sklearn.metrics import roc_curve, auc

def plot_roc(y_test, y_prob):
    fpr, tpr, _ = roc_curve(y_test, y_prob)
    plt.figure(figsize=(10,8))
    plt.title("ROC curve(auc score=%.4f)"%auc(fpr,tpr))
    plt.plot(fpr, tpr, label='ROC curve (area = %0.2f)' % auc(fpr,tpr))
    plt.plot([0,1], [0,1],'r--')

y_train_prob=models[2].predict_proba(X_train)[:,1]
y_test_prob=models[2].predict_proba(X_test)[:,1]
plot_roc(y_train, y_train_prob)
plot_roc(y_test, y_test_prob)
```

[Out]





回归模型的各项参数如下。

```
[In]
1.  print(model.intercept_)  #  打印截距
2.  print(model.coef_)     # 打印模型系数
```

```
[Out]
4.6956704711911605
[ 0.0545854   0.11158486 -0.00390938]
```

根据以上参数，多元回归模型方程式表达如下。

→ $Sales_i = 4.695 + 0.054TV_i + 0.111Radio_i - 0.003Newpaper_i + e_i, i = 1,2,3...$

预测时，可以忽略残差 e_i 带来的波动。

→ $Sales_i = 4.695 + 0.054TV_i + 0.111Radio_i - 0.003Newpaper_i$

模型的对 *TV*、*Radio*、*Newspaper* 的回归系数分别为 0.054、0.111、0.003，说明 Radio 的广告支出对销售额的影响最大，TV 次之。这意味着每增加一单位的 Radio 广告支出，可带来大约 10%的销售额提升。显然，在广告投放上，应该大力增加 Radio 渠道的支出，削减甚至放弃在 Newspaper 上的投放，以获得更高的销售额。

谢谢！