In [131…

```
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.graph_objs as go
import plotly.figure_factory as ff
from plotly import tools
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
init_notebook_mode(connected=True)
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split, GridSearchCV, StratifiedKF
from imblearn.over_sampling import SMOTE
from xgboost import XGBClassifier
from sklearn.metrics import roc_auc_score, confusion_matrix, classification_repo


import gc
from datetime import datetime
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.metrics import roc_auc_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from catboost import CatBoostClassifier
from sklearn import svm
import lightgbm as lgb
from lightgbm import LGBMClassifier
import xgboost as xgb


plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False
```

In [6]:

```
#数据说明

#id：样本ID
#person_age：借款人年龄（岁）
#person_income：借款人年收入（美元）
#person_home_ownership：借款人房屋拥有情况，取值为：RENT（租房）、OWN（拥有）、MORTGA
#person_emp_length：借款人工作年限（年）
#loan_intent：贷款意图，取值包括：EDUCATION（教育）、MEDICAL（医疗）、PERSONAL（个人
#loan_grade：贷款信用等级，从A到G表示不同的信用等级，一般来说A是最好的，依次递减
#loan_amnt：贷款金额（美元）
#loan_int_rate：贷款利率（百分比）
#loan_percent_income：贷款金额占收入的比例
#cb_person_default_on_file：是否有违约记录（Y：有，N：无）
#cb_person_cred_hist_length：信用历史长度（年）
#loan_status：是否获得贷款批准（0：未获得，1：获得）
```

In [10]:

```
data_train = pd.read_csv('train.csv')
```

In [12]:

```
data_train.head()
```

Out[12]:

	id	person_age	person_income	person_home_ownership	person_emp_length	l
0	0	37	35000	RENT	0.0	E
1	1	22	56000	OWN	6.0	
2	2	29	28800	OWN	8.0	
3	3	30	70000	RENT	14.0	
4	4	22	60000	RENT	2.0	

In [14]:
```
data_train.shape
```

Out[14]:
```
(58645, 13)
```

In [16]:
```
data_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 58645 entries, 0 to 58644
Data columns (total 13 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   id                          58645 non-null  int64
 1   person_age                  58645 non-null  int64
 2   person_income               58645 non-null  int64
 3   person_home_ownership       58645 non-null  object
 4   person_emp_length           58645 non-null  float64
 5   loan_intent                 58645 non-null  object
 6   loan_grade                  58645 non-null  object
 7   loan_amnt                   58645 non-null  int64
 8   loan_int_rate               58645 non-null  float64
 9   loan_percent_income         58645 non-null  float64
 10  cb_person_default_on_file   58645 non-null  object
 11  cb_person_cred_hist_length  58645 non-null  int64
 12  loan_status                 58645 non-null  int64
dtypes: float64(3), int64(6), object(4)
memory usage: 5.8+ MB
```

In [18]:
```
data_train.describe()
```

Out[18]:

	id	person_age	person_income	person_emp_length	loan_an
count	58645.000000	58645.000000	5.864500e+04	58645.000000	58645.000
mean	29322.000000	27.550857	6.404617e+04	4.701015	9217.556
std	16929.497605	6.033216	3.793111e+04	3.959784	5563.807
min	0.000000	20.000000	4.200000e+03	0.000000	500.000
25%	14661.000000	23.000000	4.200000e+04	2.000000	5000.000
50%	29322.000000	26.000000	5.800000e+04	4.000000	8000.000
75%	43983.000000	30.000000	7.560000e+04	7.000000	12000.000
max	58644.000000	123.000000	1.900000e+06	123.000000	35000.000

In [20]:
```
print(data_train.isnull().sum())
```

```
id                            0
person_age                    0
person_income                 0
person_home_ownership         0
person_emp_length             0
loan_intent                   0
loan_grade                    0
loan_amnt                     0
loan_int_rate                 0
loan_percent_income           0
cb_person_default_on_file     0
cb_person_cred_hist_length    0
loan_status                   0
dtype: int64
```

In [22]:
```
print(data_train.duplicated().sum())
```

```
0
```

In [24]:
```
print(data_train.person_age.value_counts().sort_index())
```

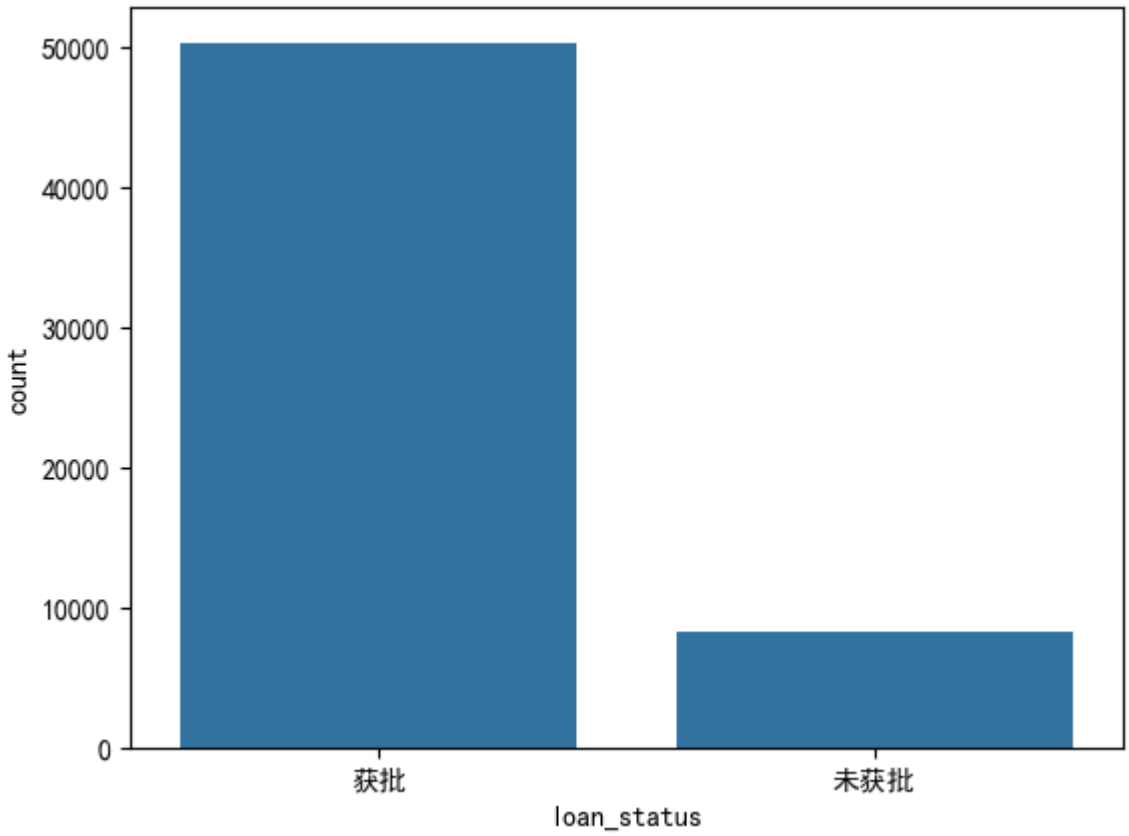
```
person_age
20       12
21     1795
22     7051
23     7726
24     6395
25     5067
26     3874
27     4450
28     3707
29     3270
30     2333
31     1917
32     1565
33     1306
34     1041
35      862
36     1117
37      992
38      745
39      536
40      438
41      433
42      291
43      320
44      229
45      163
46      164
47      125
48       97
49       59
50       63
51       69
52       62
53       75
54       60
55       34
56       29
57       25
58       35
59        6
60       28
61       13
62        7
64       10
65       13
66       11
69        6
70       10
73        3
76        1
80        2
84        2
123       1
Name: count, dtype: int64
```

In [26]:
```
print(data_train.cb_person_cred_hist_length.value_counts().sort_index())
```

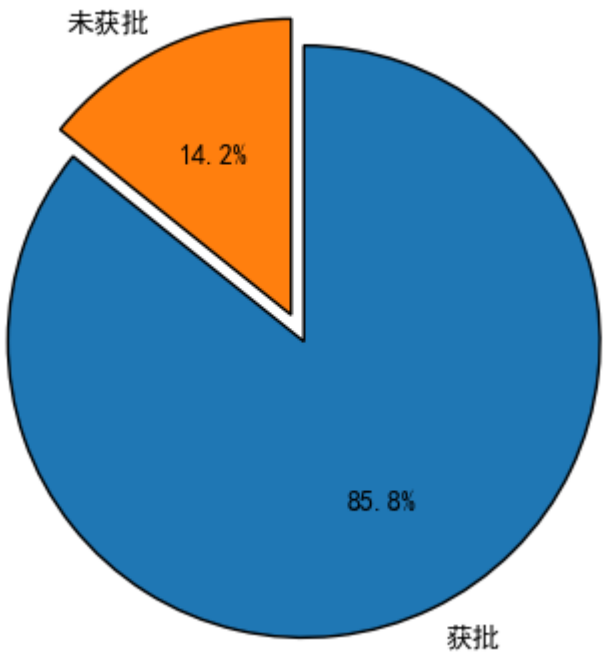
```
cb_person_cred_hist_length
2     10657
3     10708
4     10566
5      3345
6      3391
7      3392
8      3477
9      3499
10     3364
11      858
12      883
13      850
14      927
15      735
16      776
17      725
18       24
19       47
20       62
21       37
22       38
23       35
24       48
25       31
26       31
27       46
28       39
29       26
30       28
Name: count, dtype: int64
```

In [32]:
```
print(data_train["loan_status"].map({1:'未获批',0:'获批'}).value_counts())
sns.countplot(x = data_train["loan_status"].map({1:'未获批',0:'获批'}) , data = c
plt.show()
```

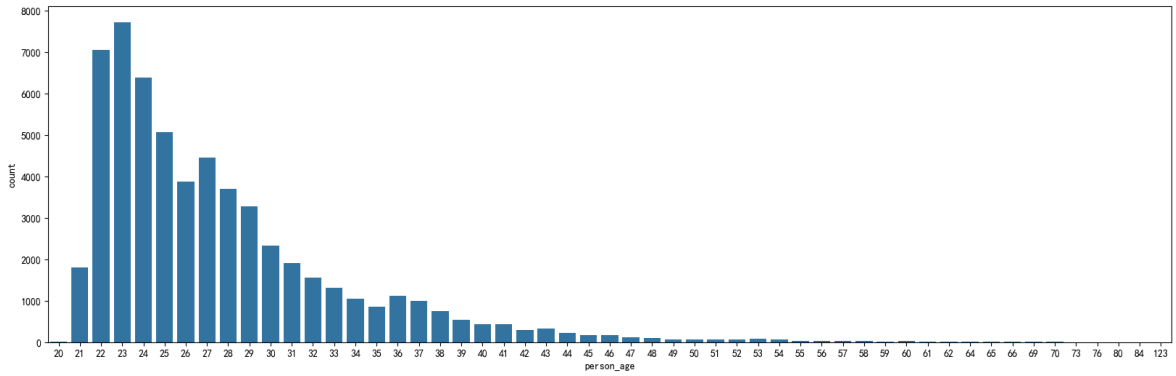
```
loan_status
获批     50295
未获批     8350
Name: count, dtype: int64
```



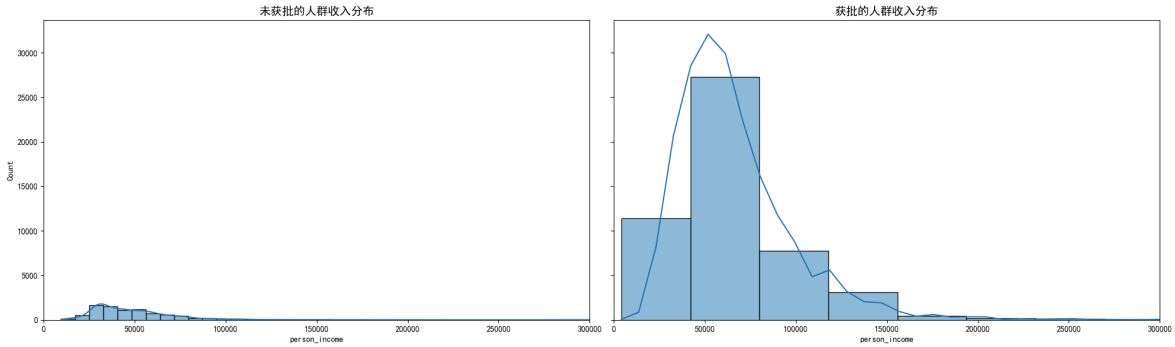
In [34]:
```
counts = data_train['loan_status'].value_counts()

plt.pie(x=counts,
        labels = counts.index.map({1: '未获批', 0: '获批'}) ,
        autopct='%1.1f%%',
        explode=[0.1, 0],
        startangle=90,
        counterclock=False,
        wedgeprops={'linewidth': 1, 'edgecolor': 'black'})
plt.show()
```



In [36]:
```
plt.figure(figsize=(20,6))
sns.countplot(x='person_age',data=data_train)
plt.show()
```



In [38]:
```
f, ax = plt.subplots(1, 2, figsize=(20, 6), sharey=True)

x_limits = (0, 300000)

sns.histplot(data_train[data_train["loan_status"] == 1]["person_income"],
             kde=True, bins=50, ax=ax[0])
ax[0].set_title('未获批的人群收入分布', fontsize=14)
ax[0].set_xlim(x_limits)

sns.histplot(data_train[data_train["loan_status"] == 0]["person_income"],
             kde=True, bins=50, ax=ax[1])
ax[1].set_title('获批的人群收入分布', fontsize=14)
ax[1].set_xlim(x_limits)

for a in ax:
    a.set_xlabel("person_income")
    a.set_ylabel("Count")

plt.tight_layout()
plt.show()
```

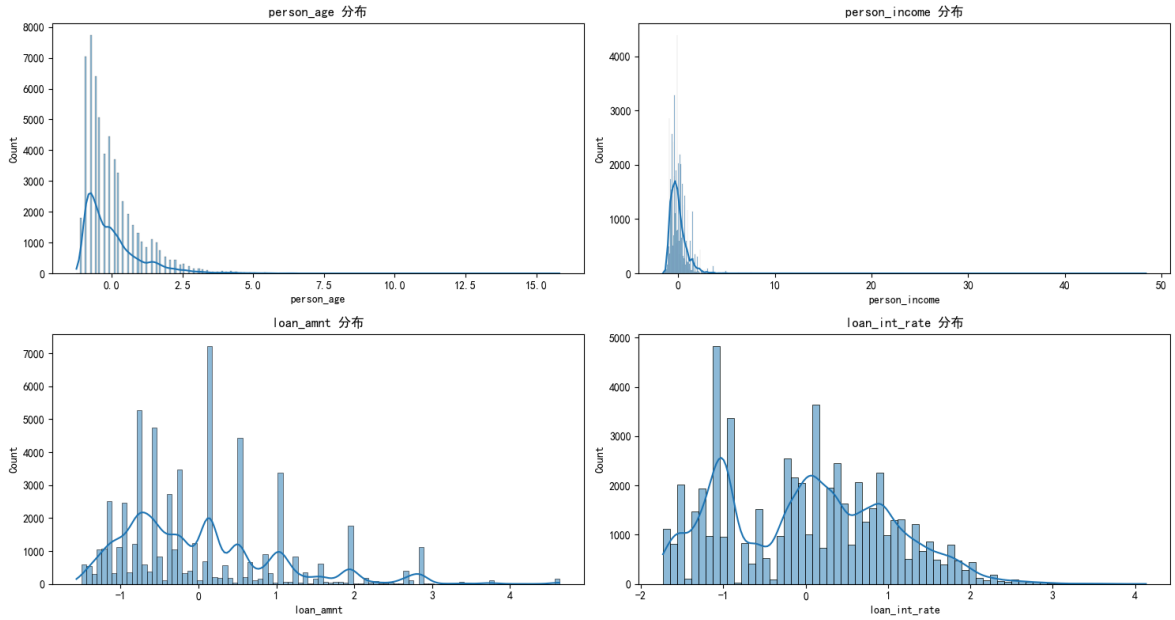


In [67]:
```
#查看类别特征分布情况

categorical_cols = ['person_home_ownership', 'loan_intent', 'loan_grade', 'cb_pe
for col in categorical_cols:
    print(f"\n{col} 分布情况: ")
    print(data_train[col].value_counts())
```

```
person_home_ownership 分布情况:
person_home_ownership
3    30594
0    24824
2     3138
1       89
Name: count, dtype: int64

loan_intent 分布情况:
loan_intent
1    12271
3    10934
4    10016
5    10011
0     9133
2     6280
Name: count, dtype: int64

loan_grade 分布情况:
loan_grade
0    20984
1    20400
2    11036
3     5034
4     1009
5      149
6       33
Name: count, dtype: int64

cb_person_default_on_file 分布情况:
cb_person_default_on_file
0    49943
1     8702
Name: count, dtype: int64
```

In [83]:
```
#查看数值特征分布情况

num_features = ['person_age', 'person_income', 'loan_amnt', 'loan_int_rate']

plt.figure(figsize=(15, 8))
for i, feature in enumerate(num_features):
    plt.subplot(2, 2, i+1)
```

```
    sns.histplot(data_train[feature], kde=True)
    plt.title(f'{feature} 分布')
plt.tight_layout()
plt.show()
```



In [69]:
```
#标准化数值特征

scaler = StandardScaler()
num_features = ['person_age', 'person_income', 'person_emp_length', 'loan_amnt',
data_train[num_features] = scaler.fit_transform(data_train[num_features])
```

In [71]:
```
#标准化类别特征

cat_features = ['person_home_ownership', 'loan_intent', 'loan_grade', 'cb_person

# Label Encoding
for col in cat_features:
    le = LabelEncoder()
    data_train[col] = le.fit_transform(data_train[col])
```

In [75]:
```
#分离特征和标签，查看分布情况

X = data_train.drop(['loan_status'], axis=1)
y = data_train['loan_status']

print("\n特征维度：", X.shape)
print("标签分布：")
print(y.value_counts(normalize=True))
```

```
特征维度： (58645, 12)
标签分布：
loan_status
0    0.857618
1    0.142382
Name: proportion, dtype: float64
```

In [79]:
```
#划分训练集和测试集，查看样本分布情况

X_train_full, X_temp, y_train_full, y_temp = train_test_split(X, y, test_size=0.
X_valid, X_test, y_valid, y_test = train_test_split(X_temp, y_temp, test_size=0.
```

```
print(f"训练集规模：{X_train_full.shape}")
print(f"验证集规模：{X_valid.shape}")
print(f"测试集规模：{X_test.shape}")

print("\n训练集标签分布：")
print(y_train_full.value_counts(normalize=True))

print("\n验证集标签分布：")
print(y_valid.value_counts(normalize=True))

print("\n测试集标签分布：")
print(y_test.value_counts(normalize=True))
```

```
训练集规模：(46916, 12)
验证集规模：(5864, 12)
测试集规模：(5865, 12)

训练集标签分布：
loan_status
0    0.857618
1    0.142382
Name: proportion, dtype: float64

验证集标签分布：
loan_status
0    0.857606
1    0.142394
Name: proportion, dtype: float64

测试集标签分布：
loan_status
0    0.85763
1    0.14237
Name: proportion, dtype: float64
```

In [81]:
```
#用SMOTE平衡样本

print("\n应用SMOTE进行过采样...")

smote = SMOTE(random_state=42)
X_train, y_train = smote.fit_resample(X_train_full, y_train_full)

print("SMOTE后训练集标签分布：")
print(y_train.value_counts(normalize=True))
```

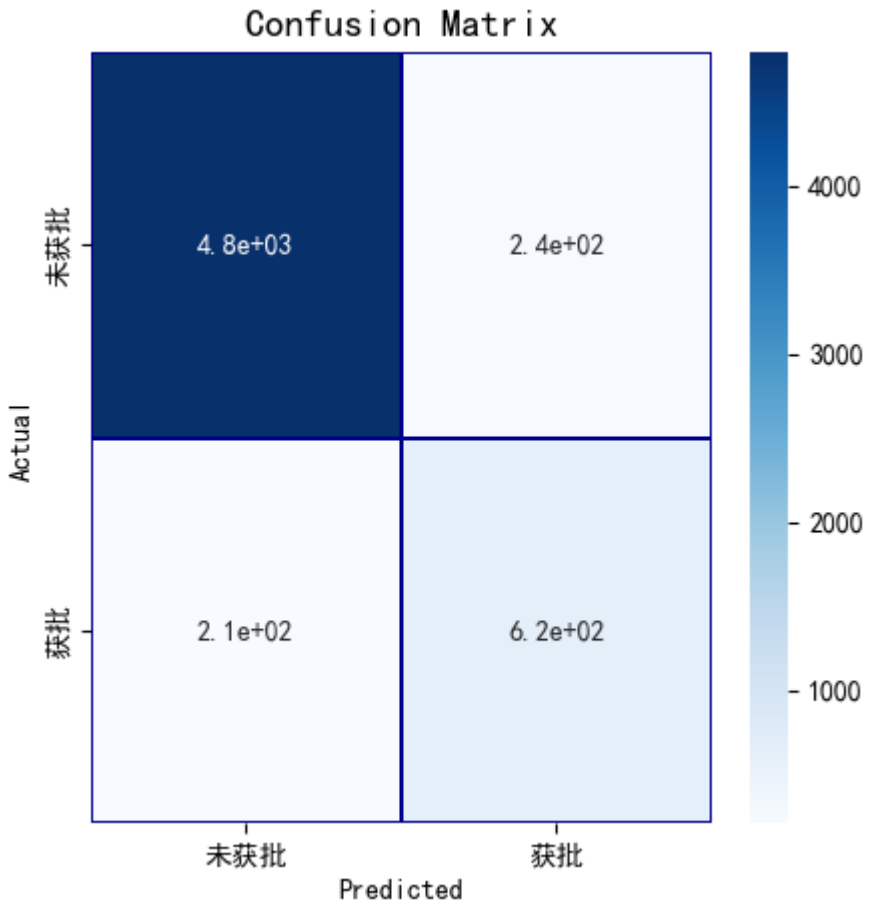
```
应用SMOTE进行过采样...
SMOTE后训练集标签分布：
loan_status
0    0.5
1    0.5
Name: proportion, dtype: float64
```

In [89]:
```
def evaluate_model(model, X_valid, y_valid, model_name):
    y_pred_prob = model.predict_proba(X_valid)[:,1]
    auc_score = roc_auc_score(y_valid, y_pred_prob)
    print(f"{model_name} 验证集 AUC: {auc_score:.4f}")
    return auc_score
```

In [119…
```
# 随机森林
rf_model = RandomForestClassifier(random_state=42, n_jobs=-1)
```

```
rf_model.fit(X_train, y_train)
rf_auc = evaluate_model(rf_model, X_valid, y_valid, "Random Forest")

preds = clf.predict(X_test)
cm = pd.crosstab(y_test , preds , rownames=['Actual'], colnames=['Predicted'])
fig, (ax1) = plt.subplots(ncols=1, figsize=(5,5))
sns.heatmap(cm,
            xticklabels=['未获批', '获批'],
            yticklabels=['未获批', '获批'],
            annot=True,ax=ax1,
            linewidths=.2,linecolor="Darkblue", cmap="Blues")
plt.title('Confusion Matrix', fontsize=14)
plt.show()
```

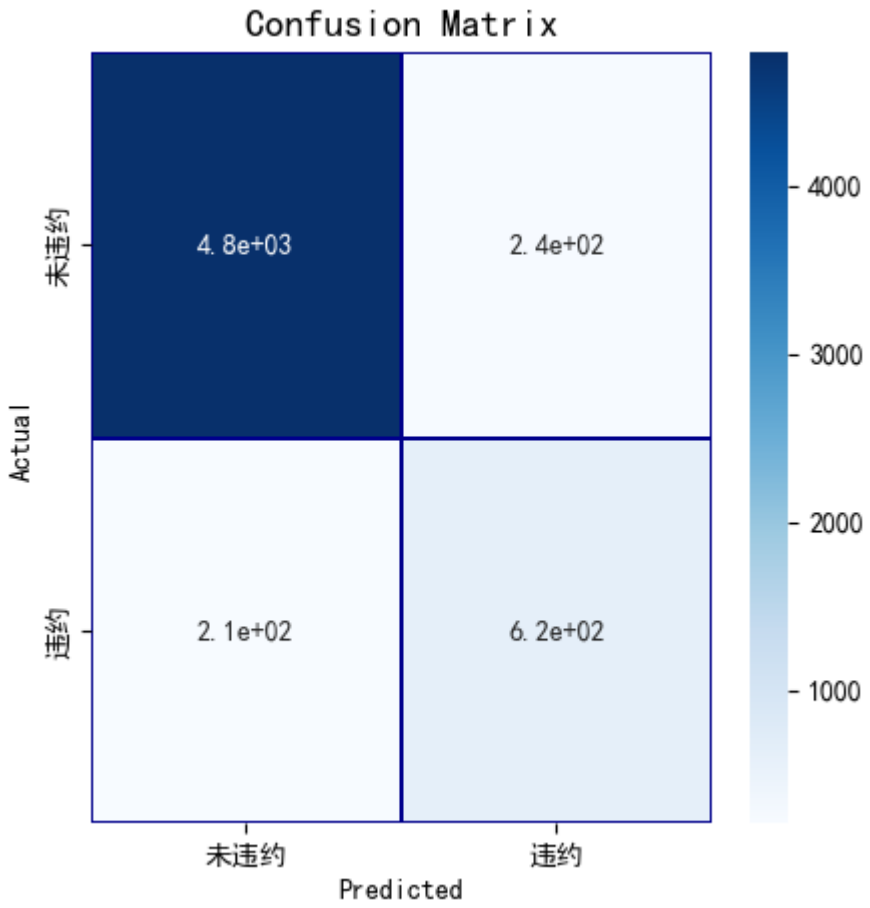
```
Random Forest 验证集 AUC: 0.9280
```



In [121…

```
#LGBM

lgbm_model = LGBMClassifier(random_state=42, n_jobs=-1)
lgbm_model.fit(X_train, y_train)
lgbm_auc = evaluate_model(lgbm_model, X_valid, y_valid, "LightGBM")

preds = clf.predict(X_test)
cm = pd.crosstab(y_test , preds , rownames=['Actual'], colnames=['Predicted'])
fig, (ax1) = plt.subplots(ncols=1, figsize=(5,5))
sns.heatmap(cm,
            xticklabels=['未违约', '违约'],
            yticklabels=['未违约', '违约'],
            annot=True,ax=ax1,
            linewidths=.2,linecolor="Darkblue", cmap="Blues")
plt.title('Confusion Matrix', fontsize=14)
plt.show()
```

```
[LightGBM] [Info] Number of positive: 40236, number of negative: 40236
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing
was 0.001111 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 2058
[LightGBM] [Info] Number of data points in the train set: 80472, number of used f
eatures: 12
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.500000 -> initscore=0.000000
LightGBM 验证集 AUC: 0.9552
```



In [127…

```python
#XGB

xgb_model = XGBClassifier(random_state=42, n_jobs=-1, use_label_encoder=False, e
xgb_model.fit(X_train, y_train)
xgb_auc = evaluate_model(xgb_model, X_valid, y_valid, "XGBoost")
```

```
C:\Users\seafarer\AppData\Roaming\Python\Python312\site-packages\xgboost\trainin
g.py:183: UserWarning:

[14:52:48] WARNING: C:\actions-runner\_work\xgboost\xgboost\src\learner.cc:738:
Parameters: { "use_label_encoder" } are not used.
```

```
XGBoost 验证集 AUC: 0.9586
```

In [133…

```python
plt.figure(figsize=(10, 8))

models = {
    "Random Forest": rf_model,
    "LightGBM": lgbm_model,
    "XGBoost": xgb_model
```

```
}

for name, model in models.items():
    y_pred_prob = model.predict_proba(X_valid)[:,1]
    fpr, tpr, _ = roc_curve(y_valid, y_pred_prob)
    roc_auc = auc(fpr, tpr)
    plt.plot(fpr, tpr, label=f'{name} (AUC = {roc_auc:.2f})')

plt.plot([0,1], [0,1], 'k--')
plt.xlim([-0.01, 1.01])
plt.ylim([-0.01, 1.01])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('模型对比 - ROC 曲线')
plt.legend(loc='lower right')
plt.grid(True)
plt.show()
```