Data Ming Report

汇报人：Liu liu,Fang Xiao,Bai Tong,Sun Penglin,YANG Chuang

1.Introduce

This report aims to analyze and predict the loan approval possibilities for loan applicants. By using machine learning techniques, especially random forest classifiers,LGBMClassified and XGBClassifier (etc.) building an effective model to help financial institutions make more informed lending decisions. The report will detail the entire process of data processing, feature selection, model building and evaluation.

2.Environment

The development environment of this Project is mainly based on the following technologies:

programming language : Python 3.8 or higher versions

operating system: Windows

To achieve data processing, model building, and evaluation, multiple Python libraries were used in the project. The following table is the main dependency libraries and their versions：

Python storeroom	versions	function
Pandas	1.3.0	data manipulation and analysis
NumPy	1.21.0	numerical calculation
Scikit-learn	0.24.0	machine learning model construction and evaluation
Matplotlib	3.4.0	Data visualization
Seaborn	0.11.0	enhance the data visualization effect

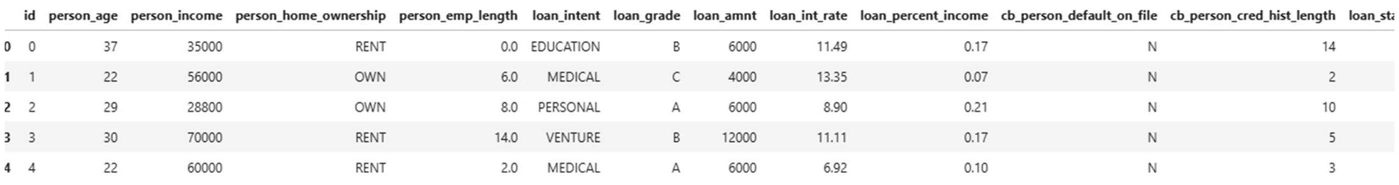
Statsmodels	0.12.0	statistical models and hypothesis testing

3.Datasets

3.1 Import data

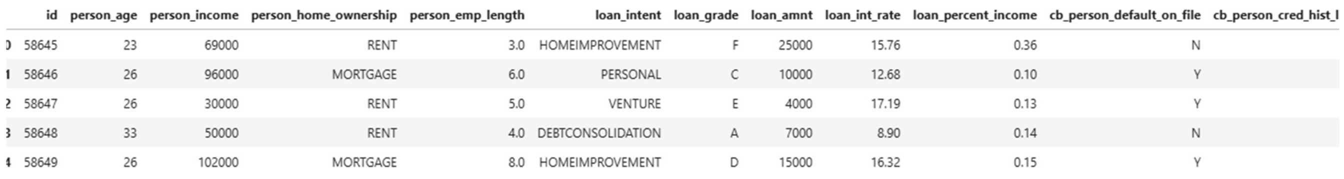
First, we import the existing training and testing datasets, and check the basic information for the dataset, including the data type, missing values, and the number of unique values

```
import pandas as pd
train = pd.read_csv("/kaggle/input/playground-series-s4e10/train.csv")
test = pd.read_csv("/kaggle/input/playground-series-s4e10/test.csv")
train.head()
test.head()
```



	id	person_age	person_income	person_home_ownership	person_emp_length	loan_intent	loan_grade	loan_amnt	loan_int_rate	loan_percent_income	cb_person_default_on_file	cb_person_cred_hist_length	loan_sta
0	0	37	35000	RENT	0.0	EDUCATION	B	6000	11.49	0.17	N	14	
1	1	22	56000	OWN	6.0	MEDICAL	C	4000	13.35	0.07	N	2	
2	2	29	28800	OWN	8.0	PERSONAL	A	6000	8.90	0.21	N	10	
3	3	30	70000	RENT	14.0	VENTURE	B	12000	11.11	0.17	N	5	
4	4	22	60000	RENT	2.0	MEDICAL	A	6000	6.92	0.10	N	3	

Figure 1. Training set



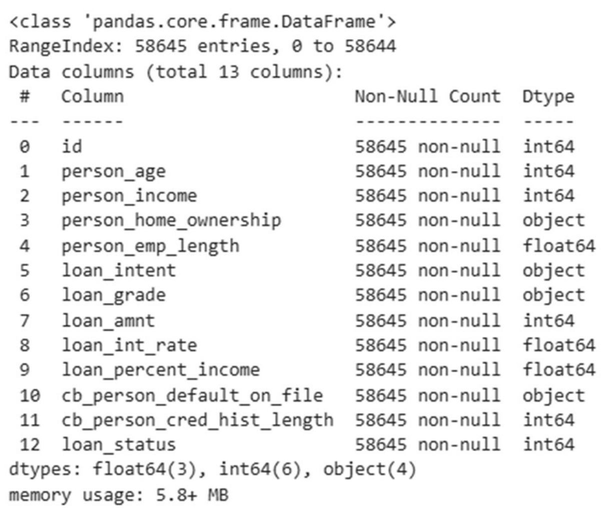
	id	person_age	person_income	person_home_ownership	person_emp_length	loan_intent	loan_grade	loan_amnt	loan_int_rate	loan_percent_income	cb_person_default_on_file	cb_person_cred_hist_l
0	58645	23	69000	RENT	3.0	HOMEIMPROVEMENT	F	25000	15.76	0.36	N	
1	58646	26	96000	MORTGAGE	6.0	PERSONAL	C	10000	12.68	0.10	Y	
2	58647	26	30000	RENT	5.0	VENTURE	E	4000	17.19	0.13	Y	
3	58648	33	50000	RENT	4.0	DEBTCONSOLIDATION	A	7000	8.90	0.14	N	
4	58649	26	102000	MORTGAGE	8.0	HOMEIMPROVEMENT	D	15000	16.32	0.15	Y	

Figure 2. Testing set

3.2 Details of the data

At this section , we examine the basic information of the dataset, including the data type, missing values, and the number of unique values. Here is the basic information for the dataset:

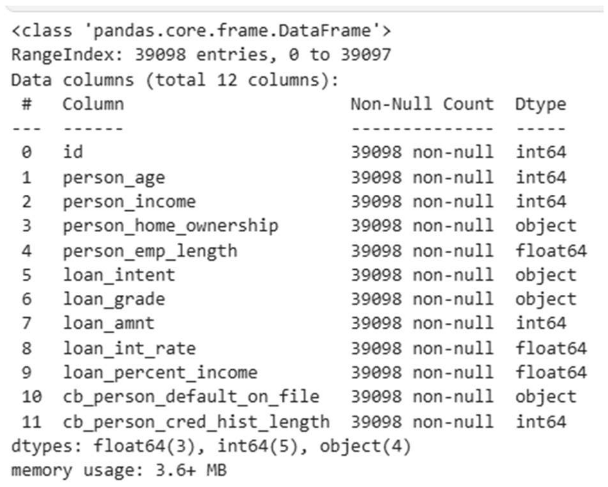
Figure 3.Details of the training data


```
<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
```

The training set contained 58,645 records and 13 features, and all features had no missing values, ensuring data integrity and reliability.

The following is the test and the information：

Figure 4.Details of the training data


```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 39098 entries, 0 to 39097
Data columns (total 12 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   id                          39098 non-null  int64  
 1   person_age                  39098 non-null  int64  
 2   person_income               39098 non-null  int64  
 3   person_home_ownership       39098 non-null  object 
 4   person_emp_length           39098 non-null  float64
 5   loan_intent                 39098 non-null  object 
 6   loan_grade                  39098 non-null  object 
 7   loan_amnt                   39098 non-null  int64  
 8   loan_int_rate               39098 non-null  float64
 9   loan_percent_income         39098 non-null  float64
 10  cb_person_default_on_file   39098 non-null  object 
 11  cb_person_cred_hist_length  39098 non-null  int64  
dtypes: float64(3), int64(5), object(4)
memory usage: 3.6+ MB
```

The test set contained 39,908 records and 12 features and all features had no missing values, ensuring data integrity and reliability.

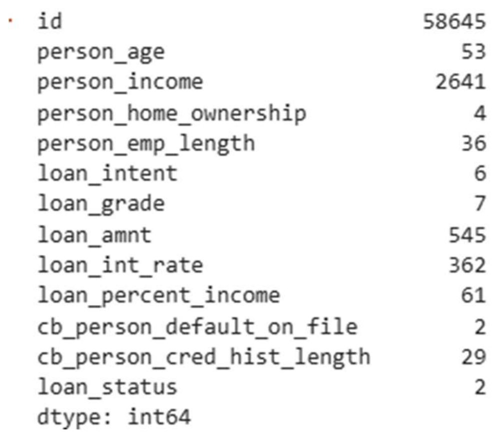
The above information characteristics explain the lender's family ownership, loan intention, loan grade, age, income, loan amount, can give us clearly show all the situation.

3.3 Assessment of the numerical features

From the previous figure, it can be seen that all the data have no missing values, which provides convenient conditions for us to better dispose of the data. The data will be further explored.

First, we need to determine the number of unique values in the dataset, which helps to understand the different values of each feature, especially for the categorical features, to judge the diversity of the features.

Figure 5.unique values



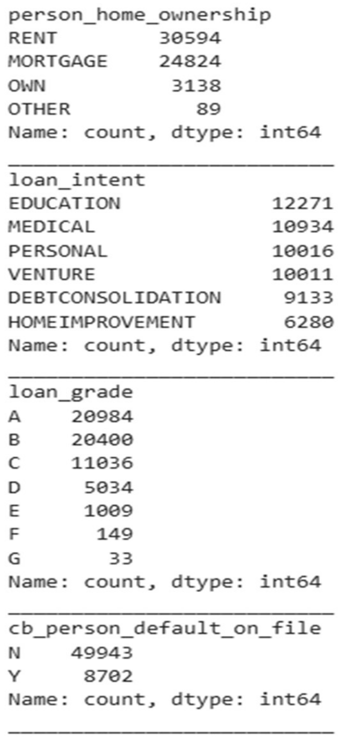
```
id                            58645
person_age                       53
person_income                  2641
person_home_ownership             4
person_emp_length                36
loan_intent                       6
loan_grade                        7
loan_amnt                       545
loan_int_rate                   362
loan_percent_income              61
cb_person_default_on_file         2
cb_person_cred_hist_length       29
loan_status                       2
dtype: int64
```

With the following code, we define a list of categorical and numerical features, which facilitates subsequent data processing and visualization work to clearly divide feature types。

```
categorical_cols=['person_home_ownership','loan_intent','loan_grade','cb_person_d
efault_on_file']
numerical_cols = ['person_age', 'person_income', 'person_emp_length', 'loan_amnt',
'loan_int_rate', 'loan_percent_income', 'cb_person_cred_hist_length']
```

Later, in order to reveal the number of samples in each category, we conducted characteristic distribution frequency statistics on the data to understand the distribution of the data.

Figure 6.frequency distribution

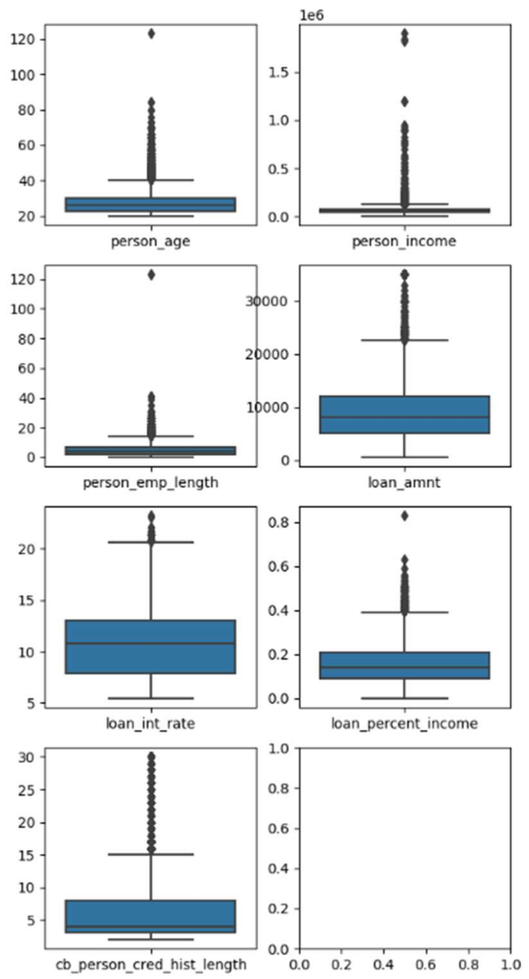


```
person_home_ownership
RENT        30594
MORTGAGE    24824
OWN          3138
OTHER          89
Name: count, dtype: int64
__________________________
loan_intent
EDUCATION            12271
MEDICAL              10934
PERSONAL             10016
VENTURE              10011
DEBTCONSOLIDATION     9133
HOMEIMPROVEMENT       6280
Name: count, dtype: int64
__________________________
loan_grade
A    20984
B    20400
C    11036
D     5034
E     1009
F      149
G       33
Name: count, dtype: int64
__________________________
cb_person_default_on_file
N    49943
Y     8702
Name: count, dtype: int64
__________________________
```

It can be found in the picture that most applicants are tenants, and the loan intention is mainly focused on education and medical care, reflecting the high loan demand. In the loan rating, most applicants are grades A, B and C, showing good credit standing, while the vast majority of applicants have no default history.

Finally, Boxplot techniques of exploratory data analysis (EDA) has employed a resistant rule for identifying possible outliers in a single batch of univariate dataset[1].We visualize the numerical features through a boxplot to assess the distribution of the data and potential outliers:

Figure 7. Data box plots



By boxplots analyzing numerical features, we found that age was mainly concentrated between 20 and 60 years, with a wide income distribution, and some applicants were significantly above average. Years of employment is mostly 0 to 10 years, indicating high mobility. The loan amount presents some outliers of high applications, the loan interest rate is relatively concentrated, and a few extreme values may reflect different credit risks. The income ratio shows that most applicants have a reasonable loan burden, while the length of the credit history is more evenly distributed

It is worth noting that a blank image appears in this figure, which may indicate that the value of this feature is very concentrated or the lack of data, resulting in the inability to effectively display its distribution, and solving the problem of this improper distribution is the problem of our future work.

4.Data preprocessing

4.1 Data cleaning

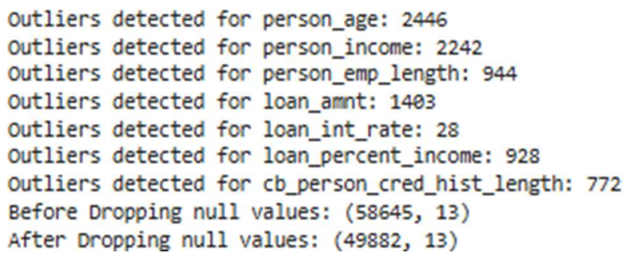
Data preprocessing includes data preparation, compounded by integration,cleaning, normalization and transformation of data; and data reduction tasks; such as feature selection, instance selection, discretization, etc[2].

We used the method of quartile distance (IQR) to effectively identify and handle outliers, by calculating the first quartile (Q1) and the third quartile (Q3) for each feature, and then calculating the upper and lower limits of outliers according to IQR (Q3-Q1)， outliers were identified by determining whether each data point was below the lower or greater than the upper limit. All records marked as outliers were then removed from the dataset and normal values were retained.

```
num_or_float_columns = train.select_dtypes(['int', 'float']).columns
columns = num_or_float_columns[1:-1]
outlier_dict = {}
new_train = train.copy()
def IQR(column):
    q1 = np.percentile(train[column], 25)
    q3 = np.percentile(train[column], 75)
    iqr = q3 - q1
    lower_fence = q1 - 1.5 * iqr
    higher_fence = q3 + 1.5 * iqr
    global new_train
    outliers = new_train[(new_train[column] < lower_fence) | (new_train[column] >
higher_fence)]
    new_train = new_train[~((new_train[column] < lower_fence) | (new_train[column] >
higher_fence))]
    outlier_dict[column] = outliers
    print(f"Outliers detected for {column}: {len(outliers)}")
```

```
for i in columns:
    IQR(i)
print("Before Dropping null values:", train.shape)
print("After Dropping null values:", new_train.shape)
```

Figure 8.detecting outliers



```
Outliers detected for person_age: 2446
Outliers detected for person_income: 2242
Outliers detected for person_emp_length: 944
Outliers detected for loan_amnt: 1403
Outliers detected for loan_int_rate: 28
Outliers detected for loan_percent_income: 928
Outliers detected for cb_person_cred_hist_length: 772
Before Dropping null values: (58645, 13)
After Dropping null values: (49882, 13)
```

As shown in Figure 8, The results of outlier detection for each numerical feature show that, The person_age has 2,446 outliers, The person_income had 2,242 outliers, The person _ emp _ length has 944 outliers, The loan_ amnt has 1403 outliers, The loan _ int _ rate has 28 outliers, The low _ percent _ income had 928 outliers, The cb _ person _ cred _ hist _ length has 772 outliers. Before removing outliers, the dataset contained 58645 records and 13 features, while the number of outliers decreased to 49882, bars, and 8756 fewer records. This process significantly improved the quality of the data and contributed to the accuracy of the subsequent analyses.

4.2 Analyze the data after cleaning

After data cleaning, we visualize the new data and draw box plots of the new data:

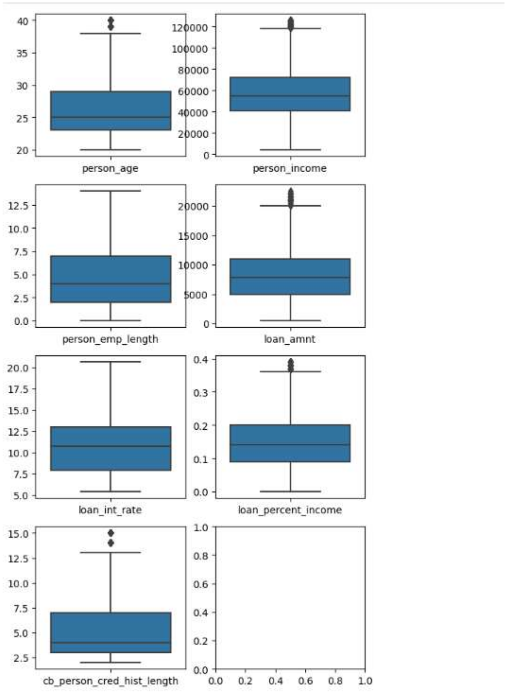


Figure 9. Data-cleaned box plots

The box plots after data cleaning showed that the distribution of the features was more reasonable and the outliers were significantly reduced. For example, the age distribution of person_age is more concentrated, the income level of person_income is clearer, and the years of employment of person _ emp _ length reflect a more realistic experience. The distribution of loan_ ammt is also more consistent, removing extremely high loans, and loan _ int _ rate is evenly distributed, making it easier to understand the loan cost after cleaning. Moreover, the distribution of percent _ income and cb _ person _ cred _ hist _ length show plausibility.

At the same time, the new box plot also appears blank, which may be a problem of missing data.

5.Prepare for modeling

5.1 heat map

After data preprocessing, Because heat maps can clearly imply aggregation or depict clusters found by statistical analysis， so we chose heat map analysis to continue a further exploration of the aspects of data features.

```
fig, ax = plt.subplots(figsize=(8,6))
dataplot = sns.heatmap(new_train.corr(numeric_only=True), cmap="YlGnBu", annot=True, ax=ax)

# Displaying heatmap
plt.show()
```

We first calculated the correlation matrix between all numerical features using the corr method in the Pandas library, setting the parameter numeric_only=True to ensure that only the correlation of the numerical features is calculated. Subsequently, the Seaborn library was used to draw the correlation heatmap, setting the graph size of 8x6 inches in the code, and selecting the yellow-green-blue color map (cmap = "YlGnBu") to enhance readability(As shown in Figure 10), while displaying the correlation value in the heatmap to quickly identify the relationships between features.

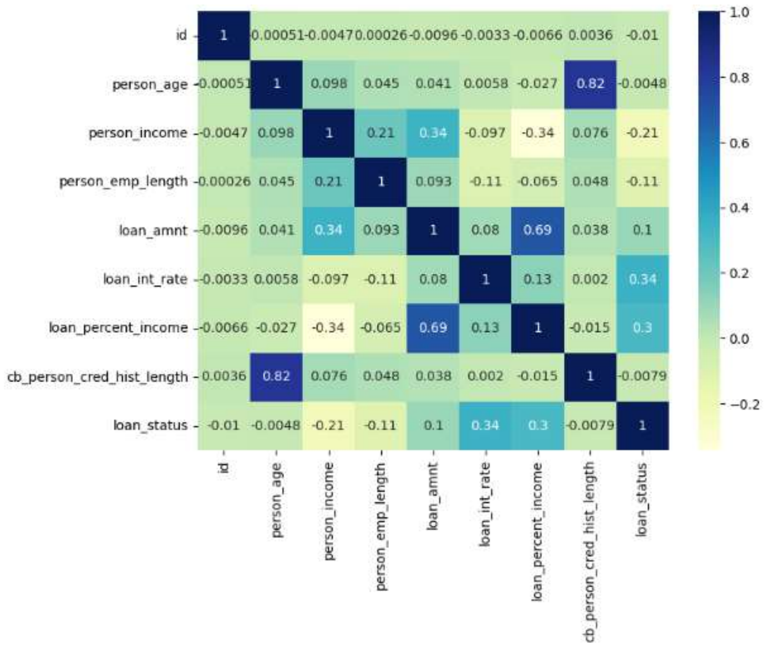


Figure 10.Data feature relationship

The correlation heat map shows the relationship between the features in the dataset, and the color depth reflects the intensity of their correlation, and the darker the color, the higher the intensity of the correlation. First, we note a significant positive correlation between person_income and loan_status,indicating that the higher the income, the greater the probability of the loan status success. Second, there is a certain positive correlation between person_mp_ length and loan_amnt, suggesting that individuals with longer employment years may also apply for a higher loan amount. In addition, a negative correlation is shown between loan_percent_income and loan_status, meaning that the higher the proportion of income to the loan, the likelihood of loan success may decrease.

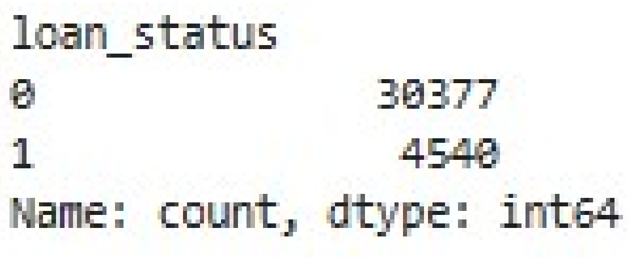
5.2 Unbalanced treatment categories

In the process of data processing, because the "id" column does not belong to the characteristics, so we need to remove the "id" column, for subsequent processing.

```
X = new_train.drop(['loan_status', 'id'], axis=1)
y = new_train['loan_status']
X.head(2)
```

The feature data set X and the target variable y were then split into training and test sets using the train_test_split function, where 70% of the data were used for training and 30% for testing. By setting random_state=5, ensure that the results of data segmentation are consistent and facilitate the recurrence of the results.

The target variable y_train is then converted to a Pandas data box, and the number of occurrences of each unique value is calculated, outputting these counts to show the distribution of the target variables in the training set(As shown in Figure 11).And judge whether there is a problem of category imbalance of the data



```
loan_status
0              30377
1               4540
Name: count, dtype: int64
```

Figure 11.Output diagram

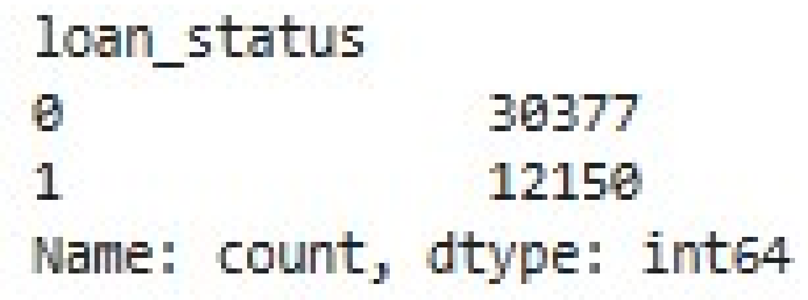
From the output, the distribution of the target variable loan _ status is as follows:

category 0,30377 times and 4540 times for category 1.

This suggests that the target variable has significant category imbalance problems, where the number of samples for category 0 is much higher than for category 1. We will use SMOTE-NC to adjust for the imbalance.If the analysis continues with imbalanced classes, it is probable that the result will demonstrate inadequate performance when forecasting new data[3].

SMOTE-NC is an oversampling technique that uses K-nearest neighbor characteristics in explanatory variables to produce synthetic data in the minority class[4].

```
from imblearn.over_sampling import SMOTENC
from imblearn.under_sampling import RandomUnderSampler
from imblearn.pipeline import Pipeline

# define pipeline
over = SMOTENC(sampling_strategy = 0.4, categorical_features=[2, 4, 5, 9],\
                    k_neighbors = 100, random_state = 42)
# transform the dataset
X_train, y_train = over.fit_resample(X_train, y_train)

print(pd.DataFrame(y_train).value_counts())
```



```
loan_status
0              30377
1              12150
Name: count, dtype: int64
```

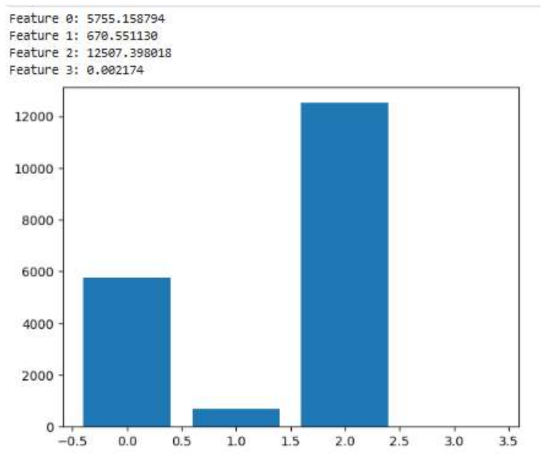
Figure11.The adjusted results

SMOTE-NC was used to deal with class imbalance in the training set. First, SMOTENC is imported from imblearn.over_sampling and a SMOTE-NC object is created, setting targets so that the number of minority class samples reaches 40% of the majority class and specifies classification features. Next, the training data is oversampled by calling the fit_resample method to generate a new balanced dataset. Finally, the output counts for each category in the processed y_train to verify the oversampling effect. After processing, the number of samples in category 1 increased from 4540 to 12150 times, although the number of samples for category 0 was still large, this indicates that SMOTE-NC effectively enhanced the minority class samples, helping to improve the predictive power and overall accuracy of the models on the minority classes.

5.3Histogram visualization

We used LabelEncoder to convert the target variables y_train and y_test to numeric formats. Next, the classified features in the training and test sets were encoded by OrdinalEncoder. Then, a feature selection function was defined and features were evaluated using the chi-square test (chi-squared) and feature selection was applied on the training data. Finally, the scores for each feature are printed and these scores are visualized by bar plots.

```
def select_features(X_train, y_train):
    fs = SelectKBest(score_func=chi2, k='all')
    fs.fit(X_train, y_train)
    X_train_fs = fs.transform(X_train)
    return X_train_fs, fs
# applying feature selection on the training data
X_train_fs, fs = select_features(X_train[categorical_cols], y_train)
# what are scores for the features
for i in range(len(fs.scores_)):
    print('Feature %d: %f' % (i, fs.scores_[i]))
# plot the scores
plt.bar([i for i in range(len(fs.scores_))], fs.scores_)
plt.show()
```



```
Feature 0: 5755.158794
Feature 1: 670.551130
Feature 2: 12507.398018
Feature 3: 0.002174
```

Figure 12. Score histogram

As can be seen from the graph, the features person_age, cb_person_cred_hist_length and cb _ person _ default _on _ file have significantly different relationships with the target variable loan _ status. The feature person_age has the highest score, indicating that it has the strongest association with loan status. In contrast, cb_person_cred_hist_length and cb _ person _ default _ on _ file had lower scores, showing a relatively weak relationship with the target variable.

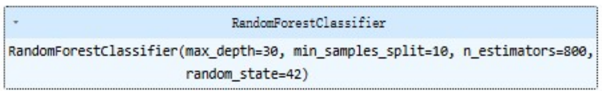
6. Utilizing bagging and boosting model

In this section, the experiment will utilize thebagging and boosting model to make predictions on the processed data. Additionally, the distribution of model prediction errors will be visualized along with the ROC curve, and the AUV value will be calculated using the roc_auc_score metric.

6.1RandomForestClassifier

The random forest uses randomization to create a large number of decision trees. The output of these trees is aggregated into a single output using voting for classification problems or averaging for regression problems[5].

First, the hyperparameters of the random forest were defined using param _ grid, including n_estimators, max_dept, min_samples_split, and min_samples_leaf.Then a random forest model is created, a RandomForestClassifier object is instantiated, and a random seed (random_state=42) is set to ensure that the results can be reproduced, and the grid search is used to optimize the model hyperparameters, and cross-validation.Finally,the best hyperparameter combination and the best cross-validation score found in the grid search are output.



```
RandomForestClassifier
RandomForestClassifier(max_depth=30, min_samples_split=10, n_estimators=800,
                       random_state=42)
```

Figure13.random forest

In the process of running, we failed to run the best number due to the large number, or the computer performance problem, which is the direction of our subsequent work.

After finding the best parameters, the code output the best parameters. Then, the model is re-instantiated based on the best parameters and trained on the training sets X_train and y_train by using the fit method. Next, the test set X_test is predicted by the predict_proba method to extract the probability value of the positive class (1). Subsequently, the false positive rate (FPR) and true positive rate (TPR) were calculated using the roc_curve function, ROC curves were plotted, and AUC values were calculated using roc_auc_score to evaluate the overall performance of the model.

```
model = RandomForestClassifier(
    max_depth=30,
    min_samples_leaf=1,
    min_samples_split=10,
    n_estimators=800,
    random_state=42
)
model.fit(X_train, y_train)
y_pred = model.predict_proba(X_test)[:, 1]
y_pred
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
plt.plot(fpr, tpr)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
```

```
# ROC Area Under Curve (AUC)

print(f'model 1 AUC score: {roc_auc_score(y_test, y_pred):.4f}')
```

```
model 1 AUC score: 0.9297
```

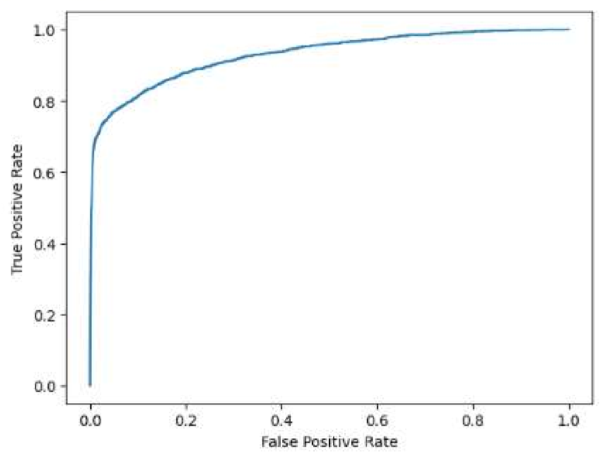
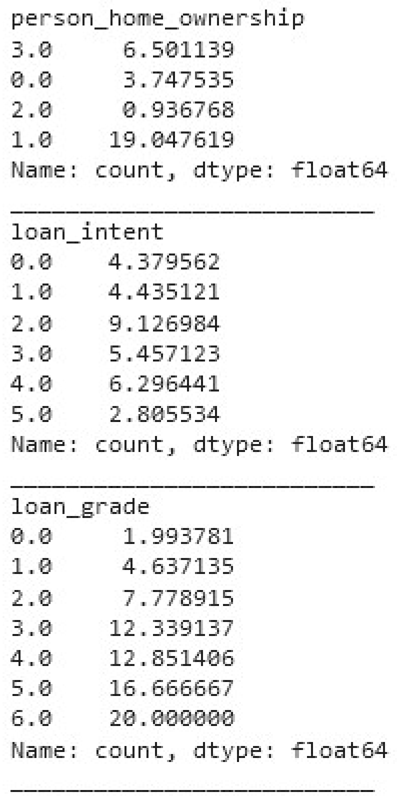


Figure14.ROC curve

This graph shows the ROC curve of the model, where the X axis represents the false positive rate (False Positive Rate), the proportion of negative classes wrongly predicted to be positive, the Y axis represents the true rate (True Positive Rate), and the proportion of positive classes correctly predicted as positive. The AUC value of the model was 0.9297, indicating its excellent performance in distinguishing positive and negative classes. The gradual upward trend of the ROC curve shows that the true rate remains high as the false positive rate increases, showing that the model is able to effectively identify positive classes in most cases.

Afterwards, the code transforms the predicted probabilities into binary results, u
sing a set threshold (0.5), and calculates the accuracy of the model on the test
set with the accuracy_score function.(Accuracy Score: 95.122 %) Next, the mi
spredicted samples are identified, and the np.logical_xor function of NumPy is
used to find predictions inconsistent with the true label y_test. Next, the relativ
e frequency of each feature value in the mispredicted sample was calculated an
d output for the specified classification features categorical_cols.

```
threshold = 0.5

y_pred = np.array((y_pred > threshold)) # 0.0 or 1.0



wrong_preds = X_test[np.logical_xor(y_test, y_pred)]

categorical_cols = ['person_home_ownership', 'loan_intent', 'loan_grade']

numerical_cols        =        ['person_income',        'person_emp_length','loan_amnt',
'loan_int_rate', 'loan_percent_income']
for c in categorical_cols:
    print(wrong_preds[c].value_counts() / X_test[c].value_counts() *100)
    print('__________________________')
```



```
person_home_ownership
3.0     6.501139
0.0     3.747535
2.0     0.936768
1.0    19.047619
Name: count, dtype: float64
__________________________
loan_intent
0.0    4.379562
1.0    4.435121
2.0    9.126984
3.0    5.457123
4.0    6.296441
5.0    2.805534
Name: count, dtype: float64
__________________________
loan_grade
0.0     1.993781
1.0     4.637135
2.0     7.778915
3.0    12.339137
4.0    12.851406
5.0    16.666667
6.0    20.000000
Name: count, dtype: float64
__________________________
```

Figure15. Variable statistics and category distribution.

The table presents the statistics and type descriptions for multiple variables. First, the person_home_ownership variable shows the category of individual home ownership, of which RENT is 6.05215, MORTGAGE is 3.69205, OWN is 1, and OTHER is 9.09909. Second, the loan _ intent variable reflects the distribution of loan intentions, including debt DEBTCONSOLIDATION is 4.314396, EDUCATIONis4.656113, household HOMEIMPROVEMENT is 2.876144, MEDICAL is 1.67772, PERSONAL is 5.67779, and VENTURE is 1. The loan grade in the loan_ grade variable shows A is 2.05529, B is 4.37973, C is 1, D is 11.92442, E is 15.49140, F is 6.89652, and missing NAN. Finally, the cb _ person _ default _ on _ file variable contains 4.110200 and the missing value N is 9.241071.

Finally, use Matplotlib to create the subplot and define a box function. Use Seaborn's boxplot function to draw box plots of the numerical features of the incorrectly predicted samples to visualize the distribution of these features.

```
fig, axes = plt.subplots(3, 2, figsize=(5*2,5*4))
def box(i):
    plot = sns.boxplot(wrong_preds[numerical_cols].iloc[:, [i]],
ax=axes[i//2][i%2])
for i in range(5):
        box(i)
plt.show()
```

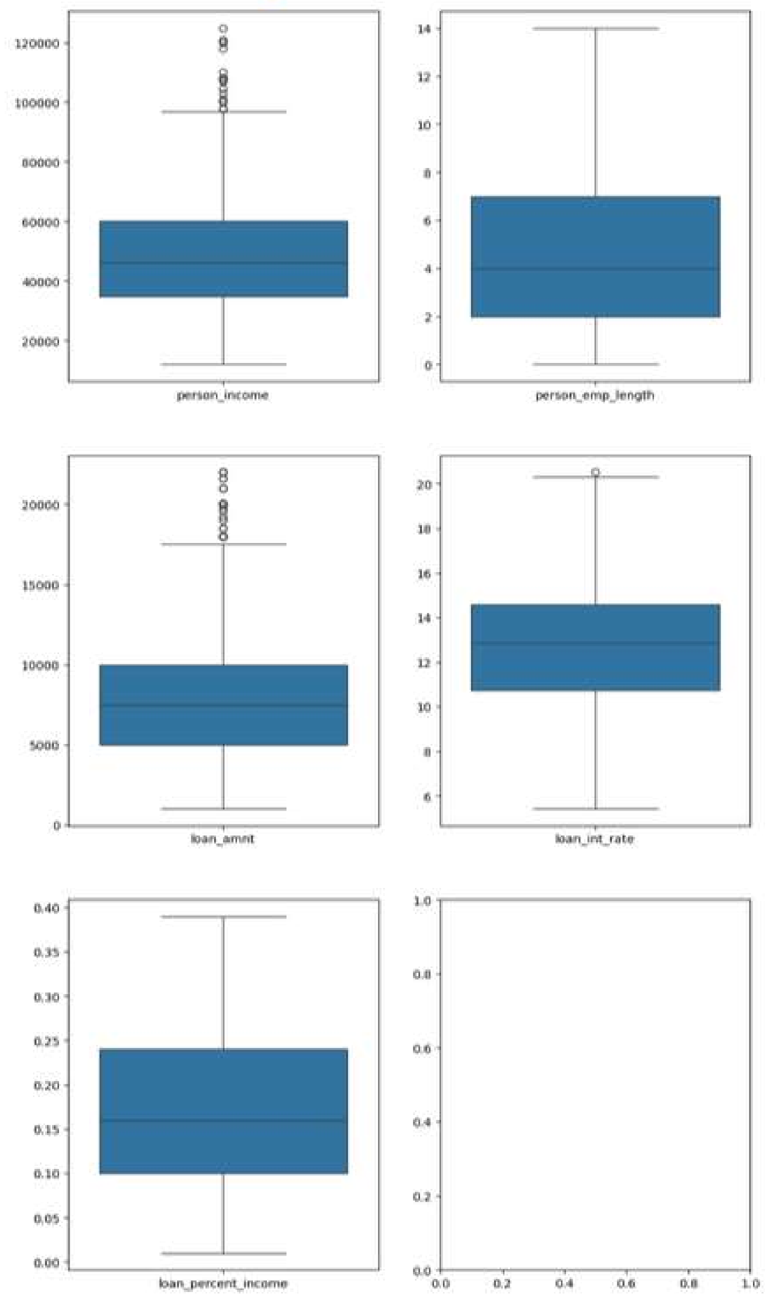


Figure.16mispredicted samples distribution

The numerical characteristics of the mispredicted samples were analyzed by box plots

person_age： The age distribution of the mispredicted sample is shown. By observing the box plot, it can be identified whether users of a certain age group are more likely to be misjudged. For example, if the median is significantly smaller, it means that young applicants may be more prone to misclassification.

person_income：Reflects the income distribution of the mispredicted sample. If the boxplots show a large income range and contain significant outliers, it may indicate a specific misclassification trend for high-or low-income applicants.

person_emp_length：Represents the distribution of working years. If the majority of mispredicted samples are found to have short working years, it may mean that applicants with lack work experience are more prone to mismisjudged.

loan_amnt：The distribution of the loan amount. If outliers exist in the box plot, it may indicate that some applicants request very high or very low loan amounts, which may affect the judgment of the model.

loan_int_rate：Distribution of lending interest rates. If a higher proportion of applicants in the mispredicted sample face high interest rates, it may indicate that applicants with risky loans are more likely to be misclassified.

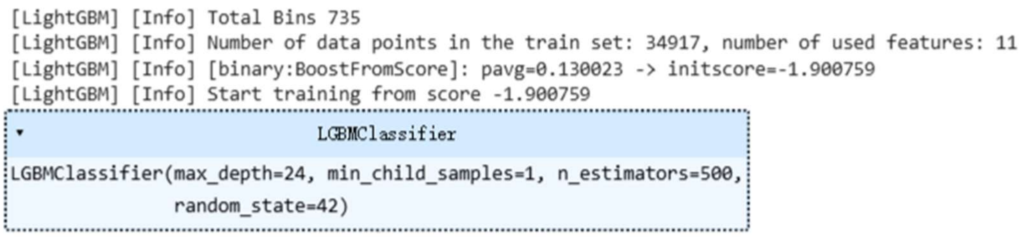
loan_percent_income：it shows the distribution of loan amount to personal income. A higher proportion may indicate that applicants are facing greater repayment pressure, and if the proportion is generally higher in the false prediction, it may reflect the inadequacy of the model in handling high-burden loan applications.

cb_person_cred_hist_length： Distribution of credit history lengths. If the proportion of short credit history in the misprediction sample is high, it may indicate that the model may bias the judgment of applicants with insufficient credit record.

6.2LGBMClassifier

LightGBM implements a highly optimized histogram-based decision tree learning algorithm, which yields great advantages on both efficiency and memory consumption[6].

First, we want to build an LGBM model, import the library, including LightGBM, sklearn evaluation indicators, and matplotlib for visualization. Then, the classification features including person_home_ownership, loan_ intent, loan_ grade, and cb _ person _ default _ on _ file are transformed into categories, so that the model can handle these variables more effectively. Then, LGBMClassifier is instantiated and multiple hyperparameters are set to ensure that the model is protected from overfitting during training. Finally, the model was fitted using the training set.



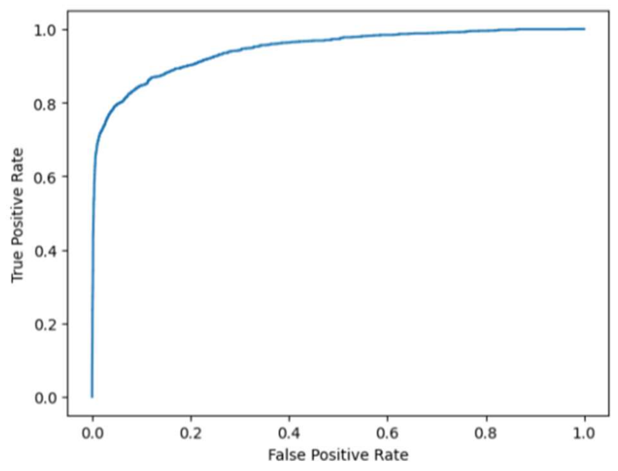
```
[LightGBM] [Info] Total Bins 735
[LightGBM] [Info] Number of data points in the train set: 34917, number of used features: 11
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.130023 -> initscore=-1.900759
[LightGBM] [Info] Start training from score -1.900759
▾                               LGBMClassifier
LGBMClassifier(max_depth=24, min_child_samples=1, n_estimators=500,
               random_state=42)
```

Key parameters of the LGBM model were successfully established including max_depth is 24, min_child_samples is 5, n_estimators is 500, and random_state is 42. The training set had 34,197 data points and used 11 features, while the training score of the model was about-1.907399, because it was evaluated using the loss function. In this case, the lower the score, the better the model usually performs.

Then we evaluated the data in the case of the model dataset again, and further analyzed the accuracy of the model by drawing the ROC curve and calculating the AUG score.

```
y_pred_lgb = model.predict_proba(X_test)[:, 1]
fpr, tpr, thresholds = roc_curve(y_test, y_pred_lgb)
plt.plot(fpr, tpr)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
print(f'model 1 AUC score: {roc_auc_score(y_test, y_pred_lgb):.4f}')
threshold = 0.5
y_pred_binary = (y_pred_lgb > threshold).astype(int)
print(f"Accuracy Score: {accuracy_score(y_test, y_pred_binary) * 100:.3f} %")
```

```
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
Text(0, 0.5, 'True Positive Rate')
```

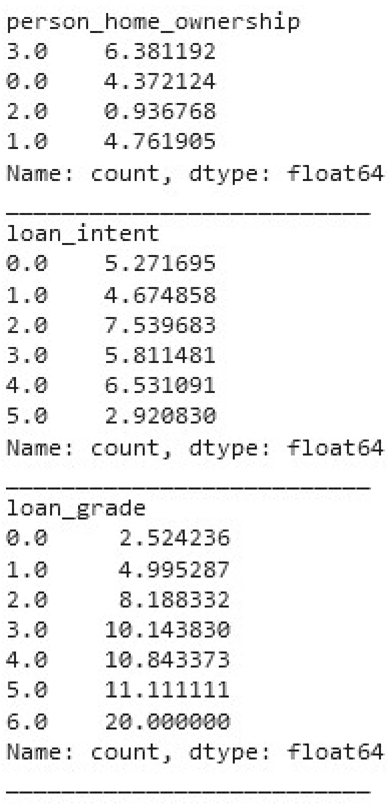


```
model 1 AUC score: 0.9447
Accuracy Score: 95.062 %
```

Figure.17 ROC and AUG

From the ROC curves, the curves are mostly close to the upper left corner, indicating that the model performs well in distinguishing between positive and negative classes. By calculating the AUC values, the ability of the model to discriminate between positive and negative class samples can be quantified. The output warning indicates no key parameters (such as num_leaves and max _ dep), which may affect the accuracy of the model. Finally, the code converted the predicted probability into two classification results by setting the threshold (0.5), and calculated the accuracy, showing that the classification effect of the model on the test set is good, with a specific accuracy of 95.062%.

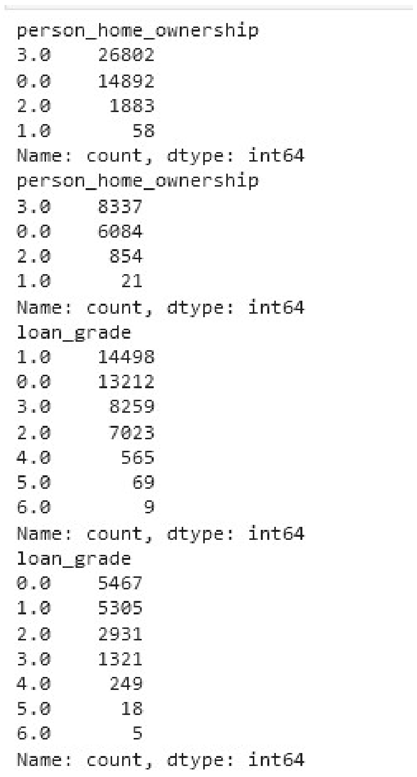
Subsequently, the model was mispredicted



```
person_home_ownership
3.0    6.381192
0.0    4.372124
2.0    0.936768
1.0    4.761905
Name: count, dtype: float64
__________________________
loan_intent
0.0    5.271695
1.0    4.674858
2.0    7.539683
3.0    5.811481
4.0    6.531091
5.0    2.920830
Name: count, dtype: float64
__________________________
loan_grade
0.0     2.524236
1.0     4.995287
2.0     8.188332
3.0    10.143830
4.0    10.843373
5.0    11.111111
6.0    20.000000
Name: count, dtype: float64
__________________________
```

Figure18.Error prediction

The analysis of the statistical results shows that the proportion of renters in the person_home_ownership feature is high, which may indicate that the model has difficulties in judging its credit risk. At the same time, the proportion of personal loan intentions such as education and medical care is also high, showing the complexity of these loan characteristics. In loan_ grade, a higher proportion of false predictions for lower grades (e. g., D) suggests that the model faces challenges in handling borrowers with low credit ratings. In addition, borrowers with previous credit default records have a large proportion of the false prediction, which may indicate that the model's ability to judge the default history is insufficient.



```
person_home_ownership
3.0    26802
0.0    14892
2.0     1883
1.0       58
Name: count, dtype: int64
person_home_ownership
3.0    8337
0.0    6084
2.0     854
1.0      21
Name: count, dtype: int64
loan_grade
1.0    14498
0.0    13212
3.0     8259
2.0     7023
4.0      565
5.0       69
6.0        9
Name: count, dtype: int64
loan_grade
0.0    5467
1.0    5305
2.0    2931
3.0    1321
4.0     249
5.0      18
6.0       5
Name: count, dtype: int64
```

Figure.19feature distribution

Based on the statistical analysis of the `person _ home _ ownership` and ` loan _ grade ` features in the training and test sets, we can see the distribution of these two features in the two data sets. First of all, in the `person _ home _ ownership` feature, the proportion of renters in the training set is significantly higher than that of other categories such as RENT: 18,765, MORTGAGE: 14,213, and the number of renters and borrowers in the test set also occupies a large proportion, indicating that the sample size of the model is relatively rich in this respect. Secondly, in the ` loan _ grade ` feature, the number of borrowers in grade C in the training set was the largest which ia7,327, while in the test set, borrowers in grades A and C also occupied A large proportion, and the number of borrowers in grades D and E was relatively small.

Finally, the boxplots are drawn against the distribution of the features

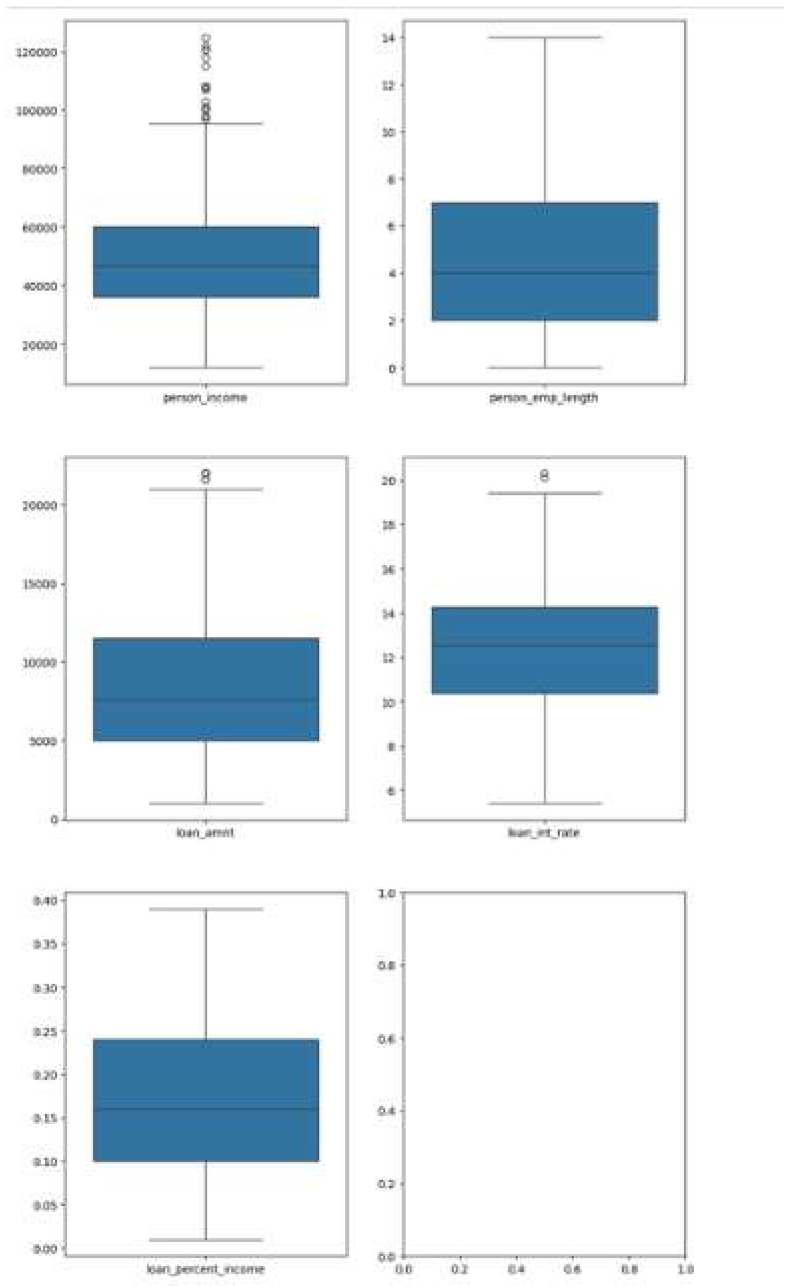


Figure20.box plot

Through the generated box plot, we can intuitively analyze the distribution of each feature. First, the boxplot of person_age shows a median around 30 years, with some outliers indicating significantly higher or lower age in some of the samples.The illustration of person_income shows that most of the borrowers' income is concentrated in the lower range, but there are also multiple outliers, possibly reflecting the presence of high income earners.person _ emp _ length shows that borrowers' employment are relatively concentrated, and that a small number of outliers may suggest an imbalance in job stability within the industry.Regarding loan_ amt, its distribution shows that most of the loan amount is within a certain range, but there are also significantly high loans.The loan _ int _ rate box chart suggests that most borrowers have lower rates, but high-risk borrowers may have higher rates.The loan_percent_income shows the proportion of loans to revenue, most borrowers, but a few high proportion of outliers. Finally, the illustration of cb _ person _ raise _ hist _ length shows that most borrowers have a long credit history, but some borrowers have a short credit history, which may affect their credit evaluation.

6.3XGBoost

XGBoost is a scalable machine learning system for tree boosting[7]. It is based on the Boosting framework, which improves the accuracy of models by gradually building a series of decision trees. XGBoost Use the gradient lifting method to optimize the loss function and improve the model performance by minimizing the prediction error.

First, the model was built using XGBClassifier and set some hyperparameters such as max _ depth ,min_child_weight, learning_rate _, and n_estimators to optimize the performance of the model. Using the enable_categorical=True option, it enables the model to handle categorical features. After the model training, we generated the predicted probabilities of the test set and calculated the false positive rate (FPR) and true positive rate (TPR) using the ROC curve function. The drawn ROC curves demonstrate the classification performance at different thresholds, with the X axis representing the false positive rate and the Y axis representing the true positive rate.

```
Text(0, 0.5, 'True Positive Rate')
```

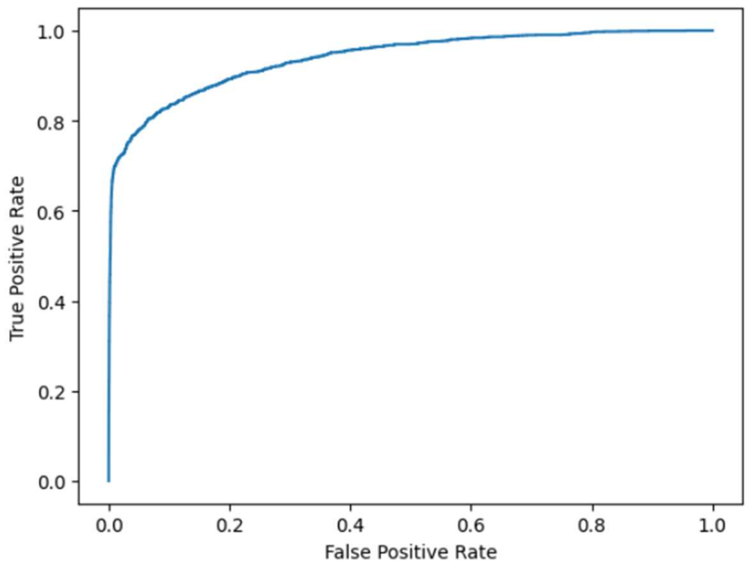
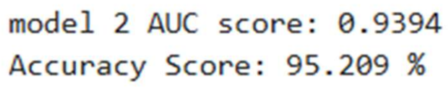


Figure21.ROC curve

From the output ROC curve, it is close to the upper left corner, indicating that the model performs well in distinguishing between positive and negative samples. The closer the curve is to the top left corner, the better the classification performance of the model is.

```
#xgb
import xgboost as xgb
from sklearn.metrics import roc_curve, roc_auc_score, accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
model = xgb.XGBClassifier(max_depth=24, min_child_weight=1, learning_rate=0.1, n_estimators=500, random_state=42, enable_categorical=True)
#cuz xgb model do not support the categorical type, using 'enable_categorical=True' to make sure it can go on.
model.fit(X_train, y_train)
y_pred_xgb = model.predict_proba(X_test)[:, 1]
fpr, tpr, thresholds = roc_curve(y_test, y_pred_xgb)
plt.plot(fpr, tpr)
plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')
```

AUG score to view the model accuracy:

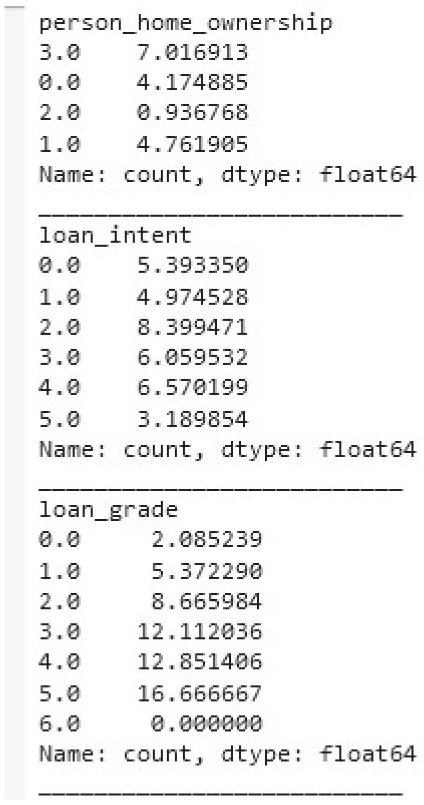


```
model 2 AUC score: 0.9394
Accuracy Score: 95.209 %
```

Figure22.result

Based on the evaluation results of the model, the AUC score was 0.9394, close to 1, indicating that the model performs well in distinguishing positive and negative samples. Moreover, the model achieved an accuracy of 95.209% on the test set, showing its ability to be correctly classified in most cases.

Next up is the error prediction:



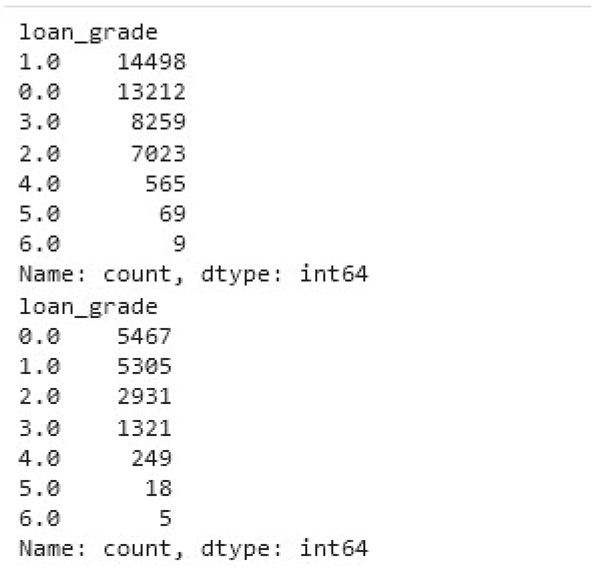
```
person_home_ownership
3.0    7.016913
0.0    4.174885
2.0    0.936768
1.0    4.761905
Name: count, dtype: float64
___________________________
loan_intent
0.0    5.393350
1.0    4.974528
2.0    8.399471
3.0    6.059532
4.0    6.570199
5.0    3.189854
Name: count, dtype: float64
___________________________
loan_grade
0.0     2.085239
1.0     5.372290
2.0     8.665984
3.0    12.112036
4.0    12.851406
5.0    16.666667
6.0     0.000000
Name: count, dtype: float64
___________________________
```

Figure23.Error prediction

In person_home_ownership, the proportion of false predictions for RANT was 5.99%, 3.90% for MORTGAGE, and 1.77% for OWN, compared with 4.55% for OTHER. In terms of loan_ intent, the proportion of false predictions in DEBTCONSOLIDATION was 5.08%, EDUCATION was 4.29%, and

HOMEIMPROVEMENT was 6.00%, while MEDICAL had no false predictions. For loan_ grade, the proportion of false predictions was 2.18% for A, 4.50% for B, 10.04% for C, 12.80% for D and 6.90% for E, while F and G had no false predictions. In cb _ person _ default _ on _ file, the proportion of false predictions for N was 4.09% and for Y was 8.79%. These results reveal differences in model performance in specific categories and provide a valuable reference for subsequent model optimization.

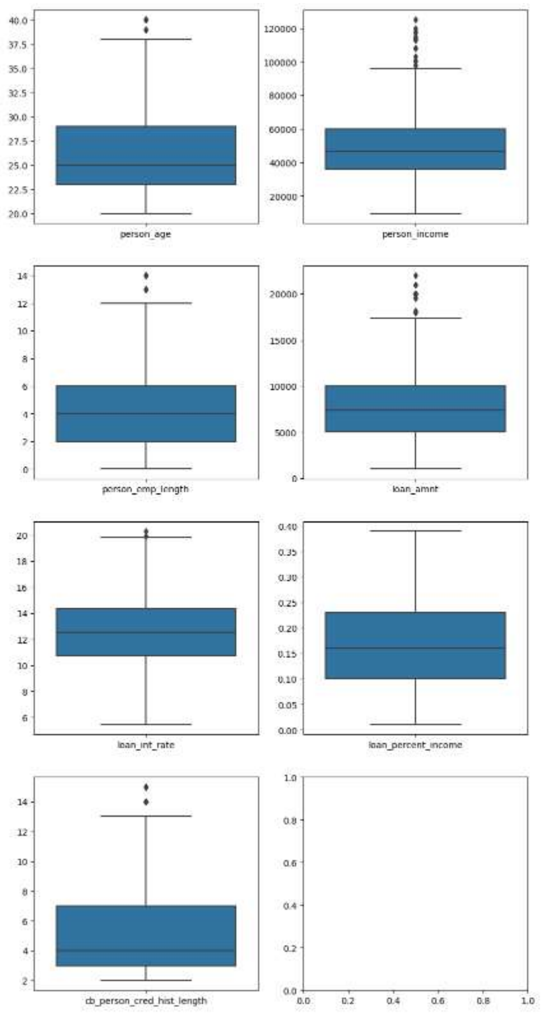
Then a summary of the category distribution of lan _ grade features in the training and test sets:



```
loan_grade
1.0    14498
0.0    13212
3.0     8259
2.0     7023
4.0      565
5.0       69
6.0        9
Name: count, dtype: int64
loan_grade
0.0    5467
1.0    5305
2.0    2931
3.0    1321
4.0     249
5.0      18
6.0       5
Name: count, dtype: int64
```

As can be seen from the figure, there are 12,737 Class A loans, 11,993 Class B loans, 6,760 Class C loans, 2,282 Class D loans, 539 Class E loans, 56 Class F loans, and 10 Class G loans. It is shown that Most loans are concentrated on high ratings. In the test set, there were 5,498 grade A loans, 5,183 grade B loans, 2,778 grade C loans, 1,216 grade D loans, 259 grade E loans, 29 grade F loans, and only 2 grade G loans. Although the two maintain a certain consistency in category distribution, the number of samples in each category in the test set is relatively small, especially for F and G-level loans, which may affect the training and evaluation results of the model.

Finally, we boxplot the distribution of each feature



According to the boxplot analysis, person_age has a wide age distribution with some outliers, indicating that most of the ages are concentrated in the middle range, while some individuals are older. The median of person_income is relatively high, but it also has obvious outliers, showing that some individuals earn far more than others. The distribution of person_crmp_length is relatively concentrated, with more samples near the median and fewer outliers, indicating that the credit record length of most people is relatively consistent. The loan amount of loan_ amt also has outliers, with the overall median, indicating that the loan amount is mainly concentrated at the middle level. The interest rate distribution of loan_int_rate is relatively concentrated, and the median is low, and the outliers may represent some loans with much higher interest rates than others. The loan_percent_income shows that the proportion of loans to revenue is mostly concentrated in the lower range, with outliers indicating a higher loan ratio for some individuals. Finally, the credit history length distribution of cb_person_cred_hist_length is also relatively concentrated, with fewer outliers. The analysis of these features provides an important reference for understanding the data structure and subsequent modeling.

7.Ensemble Learning

In this section, the experiment will explore how to use ensemble learning methods to train and evaluate different machine learning models. The core idea of ensemble learning is to combine the predictions of multiple models in the hope of achieving better performance than a single model. This experiment will implement this process through the model_trainer function, which accepts a model, training data X and y, test data, and the number of cross-validation splits n_splits and the random state random_state. In this experiment, we will use KNN, logistic regression, and random forest models, stack them with a Ridge Regression model, and make predictions on the preprocessed data.

7.1Define the model training and evaluation function

```
def model_trainer(model, X, y, test, n_splits=10, random_state=42):

skfold=StratifiedKFold(n_splits=n_splits,shuffle=True,random_state=random_state
)

    roc_aucs = []
    test_pred = np.zeros(len(test))
    oof_train_preds = np.zeros(len(y))

    model_name = model[-1].__class__.__name__ if isinstance(model, Pipeline)
else model.__class__.__name__
```

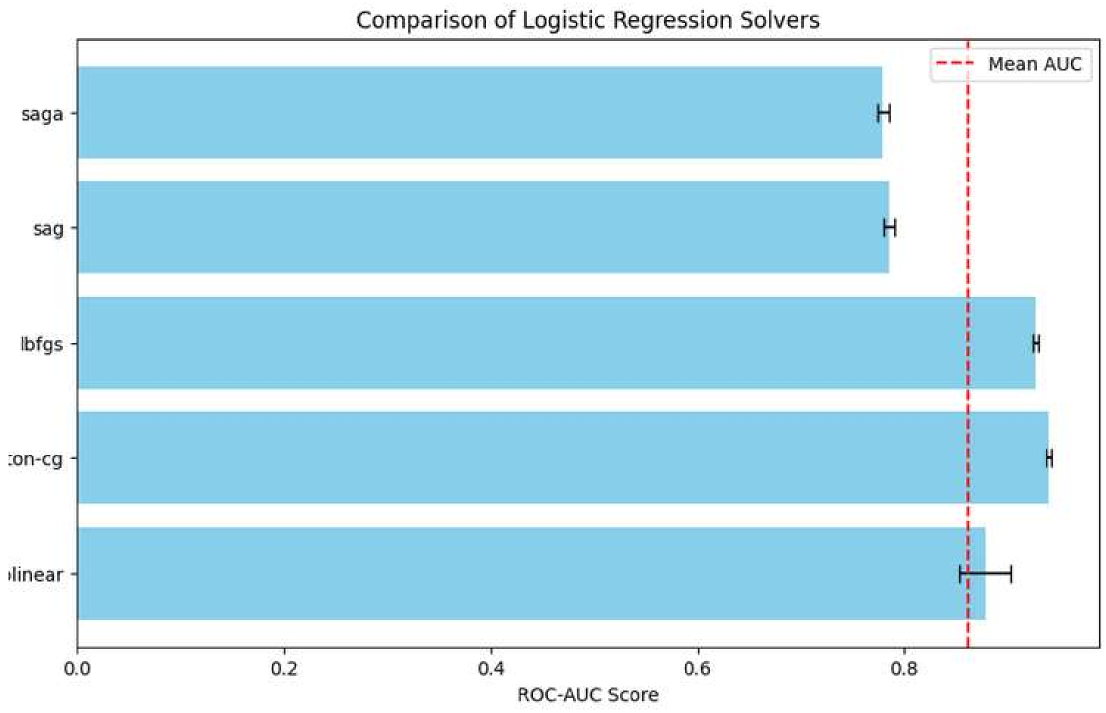
```
    print("="*72)
    print(f"Training {model_name}")
    print("="*72,sep='\n')

    for fold, (train_idx, test_idx) in enumerate(skfold.split(X, y)):
        X_train, y_train = X.iloc[train_idx, :], y[train_idx]
        X_test, y_test = X.iloc[test_idx, :], y[test_idx]

        model_clone = clone(model)
        model_clone.fit(X_train, y_train)
        try:
            y_pred_proba = model_clone.predict_proba(X_test)[:,1]
            test_pred += model_clone.predict_proba(test)[:, 1]
        except:
            y_pred_proba = model_clone.predict(X_test)
            test_pred += model_clone.predict(test)
        oof_train_preds[test_idx] = y_pred_proba
        roc_auc = roc_auc_score(y_test, y_pred_proba)
        roc_aucs.append(roc_auc)
        print(f"Fold {fold+1} --> ROC_AUC Score: {roc_auc:.6f}")

        del model_clone, X_train, X_test, y_train, y_test
        gc.collect()

    print(f"\nAverage Fold ROC_AUC Score: {np.mean(roc_aucs):.6f} \xb1
{np.std(roc_aucs):.6f}\n")
    return test_pred/skfold.get_n_splits(), oof_train_preds
```

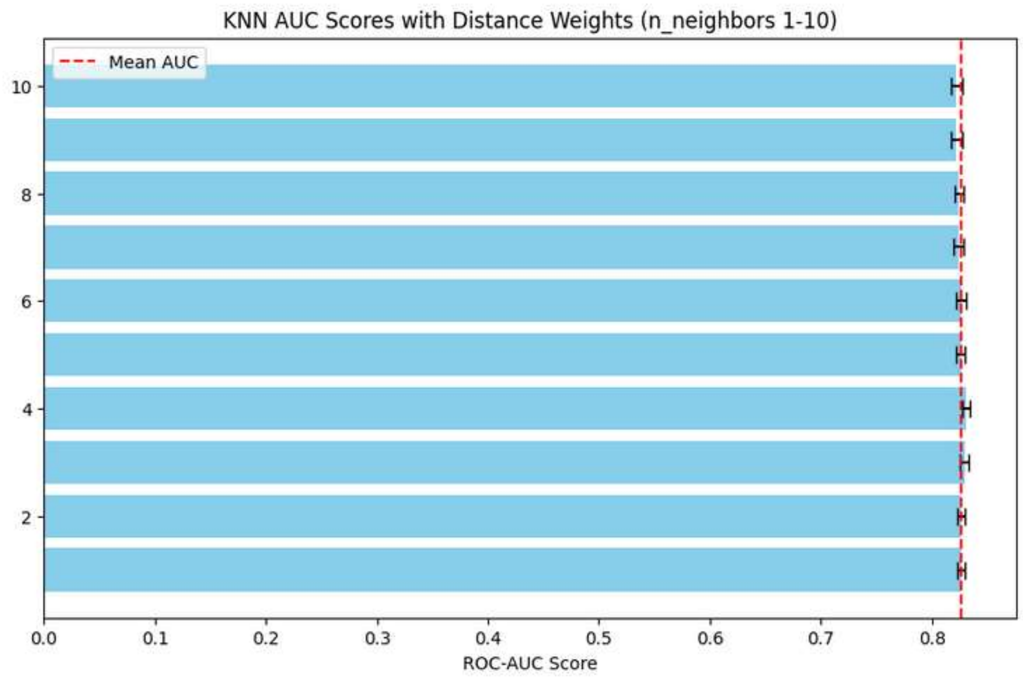
Initialize the parameters for stratified K-fold cross-validation. In each fold: divide the training set and the validation set. Clone the model and train it on the training set. Make predictions on the validation set and calculate the ROC AUC score. Save the predictions for the test set and the Out-of-Fold (OOF) predictions for the training set. Return the results: the average predictions for the test set and the OOF predictions for the training set.

7.2 Model tuning and comparison

By setting different solvers and using cross-validation to evaluate the ROC AUC scores of each model, the performance of each solver is ultimately recorded. A bar chart is used to compare the average ROC AUC scores and standard deviations of different solvers, visually displaying the differences in performance among various models.

```
for solver in solvers:
    oof_aucs = []
    oof_train_preds = np.zeros(len(y))
    for fold, (train_idx, test_idx) in enumerate(skfold.split(X, y)):
        X_train, y_train = X.iloc[train_idx], y[train_idx]
        X_test, y_test = X.iloc[test_idx], y[test_idx]
        log_reg_clf = LogisticRegression(solver=solver, max_iter=1000)
        log_reg_clf.fit(X_train, y_train)
        y_pred = log_reg_clf.predict_proba(X_test)[:, 1]
        auc = roc_auc_score(y_test, y_pred)
        oof_aucs.append(auc)
        oof_train_preds[test_idx] = y_pred
```

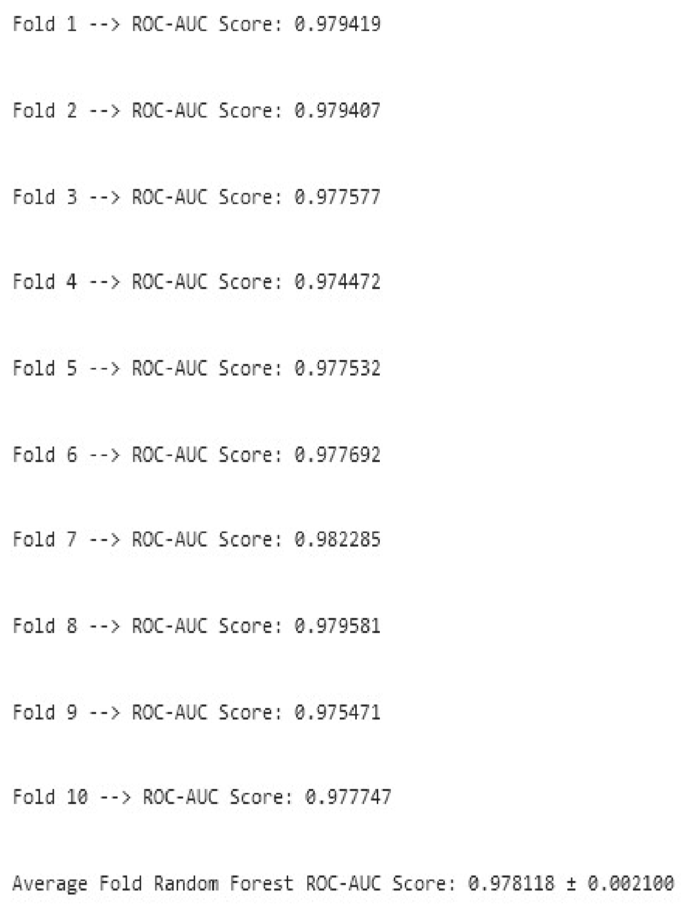




7.3 Implement model training and evaluation

After selecting the solver with the highest AUC score, train and evaluate the logistic regression model through K-fold cross-validation. First, set appropriate hyperparameters, then in each fold, divide the training set and the validation set, train the model, and calculate the ROC AUC score for the validation set. Finally, summarize the scores from all folds, output the average score and standard deviation, and save the OOF prediction results for the training set and the average prediction results for the test set.

```
    oof_preds = []
    oof_aucs = []
    oof_train_preds = np.zeros(len(y))
    for fold, (train_idx, test_idx) in enumerate(skfold.split(X, y)):
        X_train, y_train = X.iloc[train_idx], y[train_idx]
        X_test, y_test = X.iloc[test_idx], y[test_idx]
        rf_clf = RandomForestClassifier(**rf_params)
        rf_clf.fit(X_train, y_train)
        y_pred = rf_clf.predict_proba(X_test)[:, 1]
        auc = roc_auc_score(y_test, y_pred)
        oof_aucs.append(auc)
        oof_train_preds[test_idx] = y_pred
        test_preds_rf = rf_clf.predict_proba(test)[:, 1]
        oof_preds.append(test_preds_rf)
        print(f"\nFold {fold + 1} --> ROC-AUC Score: {auc:.6f}\n")
    auc_mean = np.mean(oof_aucs)
    auc_std = np.std(oof_aucs)
    print(f"\nAverage Fold RandomForest ROC-AUC Score: {auc_mean:.6f} ±
{auc_std:.6f}\n")
    train_preds['rf'] = oof_train_preds
    test_preds['rf'] = np.mean(oof_preds, axis=0)
```

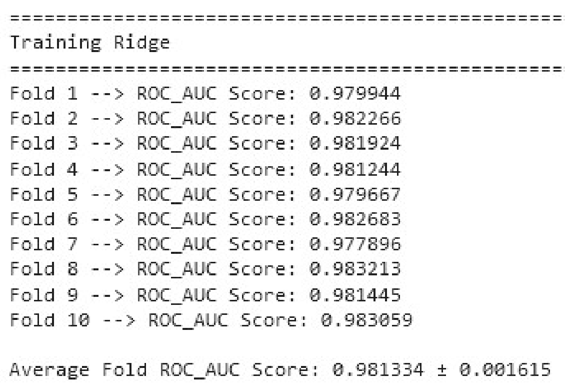


```
Fold 1 --> ROC-AUC Score: 0.979419

Fold 2 --> ROC-AUC Score: 0.979407

Fold 3 --> ROC-AUC Score: 0.977577

Fold 4 --> ROC-AUC Score: 0.974472

Fold 5 --> ROC-AUC Score: 0.977532

Fold 6 --> ROC-AUC Score: 0.977692

Fold 7 --> ROC-AUC Score: 0.982285

Fold 8 --> ROC-AUC Score: 0.979581

Fold 9 --> ROC-AUC Score: 0.975471

Fold 10 --> ROC-AUC Score: 0.977747

Average Fold Random Forest ROC-AUC Score: 0.978118 ± 0.002100
```

7.4 Stacking

Use a Ridge regression model as the final meta-learner to leverage the prediction results generated by previously trained models (such as logistic regression, KNN, etc.), thereby generating a more powerful final prediction

```
test_preds_df = pd.DataFrame(test_preds)
train_preds_df = pd.DataFrame(train_preds)
ridge = Ridge(positive=True, tol=1e-6)
ridge_pred_stack,_=model_trainer(ridge,rain_preds_df,y,test_preds_df,random_state=101)
```



```
=====================================================================
Training Ridge
=====================================================================
Fold 1 --> ROC_AUC Score: 0.979944
Fold 2 --> ROC_AUC Score: 0.982266
Fold 3 --> ROC_AUC Score: 0.981924
Fold 4 --> ROC_AUC Score: 0.981244
Fold 5 --> ROC_AUC Score: 0.979667
Fold 6 --> ROC_AUC Score: 0.982683
Fold 7 --> ROC_AUC Score: 0.977896
Fold 8 --> ROC_AUC Score: 0.983213
Fold 9 --> ROC_AUC Score: 0.981445
Fold 10 --> ROC_AUC Score: 0.983059

Average Fold ROC_AUC Score: 0.981334 ± 0.001615
```

8.conclusion

This experiment utilized machine learning techniques, including random forests, LGBM, and XGBoost classifiers, to build models that assist financial institutions in optimizing loan decisions. The experiment covered the entire process from data processing, feature selection, to model construction and evaluation. Outliers were handled using the interquartile range method, and category imbalance was addressed with the SMOTE-NC technique, followed by feature selection. Model evaluation showed that the XGBoost classifier achieved an accuracy rate as high as 95.209% on the test set. Finally, the experiment employed ensemble learning, using ridge regression as the meta-learner to integrate the prediction results of multiple models, thereby enhancing predictive performance.

9.Expectation

In this project, you can find that the random forest model on the basis of k fold cross validation will make the model run load timeout, so this is we need to solve the problem in the future, may need stronger GPU, or have updated algorithm to solve this problem, to improve the accuracy of the model, when drawing the heat map should pay attention to the number of features, reduce multicollinearity, to prevent the model overfitting.

10.Reference

[1] Babura, B. I., Adam, M. B., Samad, A. R. A., Fitrianto, A., & Yusif, B. (2018, November). Analysis and assessment of boxplot characters for extreme data. In Journal

of Physics: Conference Series (Vol. 1132, No. 1, p. 012078). IOP Publishing.

[2] García, S., Luengo, J., & Herrera, F. (2015). Data preprocessing in data mining (Vol. 72, pp. 59-139). Cham, Switzerland: Springer International Publishing.

[3] Ratnasari, A. P. (2024). Performance of Random Oversampling, Random Undersampling, and SMOTE-NC Methods in Handling Imbalanced Class in Classification Models. Valley International Journal Digital Library, 494-501.

[4] Q. H. Doan, S. H. Mai, Q. T. Do, and D. K. Thai, “A cluster-based data splitting method for small sample and class imbalance problems in impact damage classification,”Appl.Soft Comput.,vol.120,p.08628,2022,doi: 10.1016/j.asoc.2022.108628.

[5] Rigatti, S. J. (2017). Random forest. Journal of Insurance Medicine, 47(1), 31-39.

[6] "Features — LightGBM 3.1.0.99 documentation". lightgbm.readthedocs.io.

[7] Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, p.785-794

[8] Tolles, Juliana; Meurer, William J (2016). "Logistic Regression Relating Patient Characteristics to Outcomes". JAMA. 316 (5): 533–4.

[9] Fix, Evelyn; Hodges, Joseph L. (1951). Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties