Credit Matching Specification

V2.0

1 Introduction

This document defines the requirements of the product. It is a reference for the developers, testers and other related people.

The product builds user portrait based on user's basic information, it uses big data and deep learning algorithms to build an advanced model for loan products, and recommends most suitable products for users.

There are two kinds of loans, credit loans and mortgage loans. The most suitable product must be recommended according to user's qualification and basic information.

2 Product Design

The product is based on a webServer framework, which provides users with standard APIs.

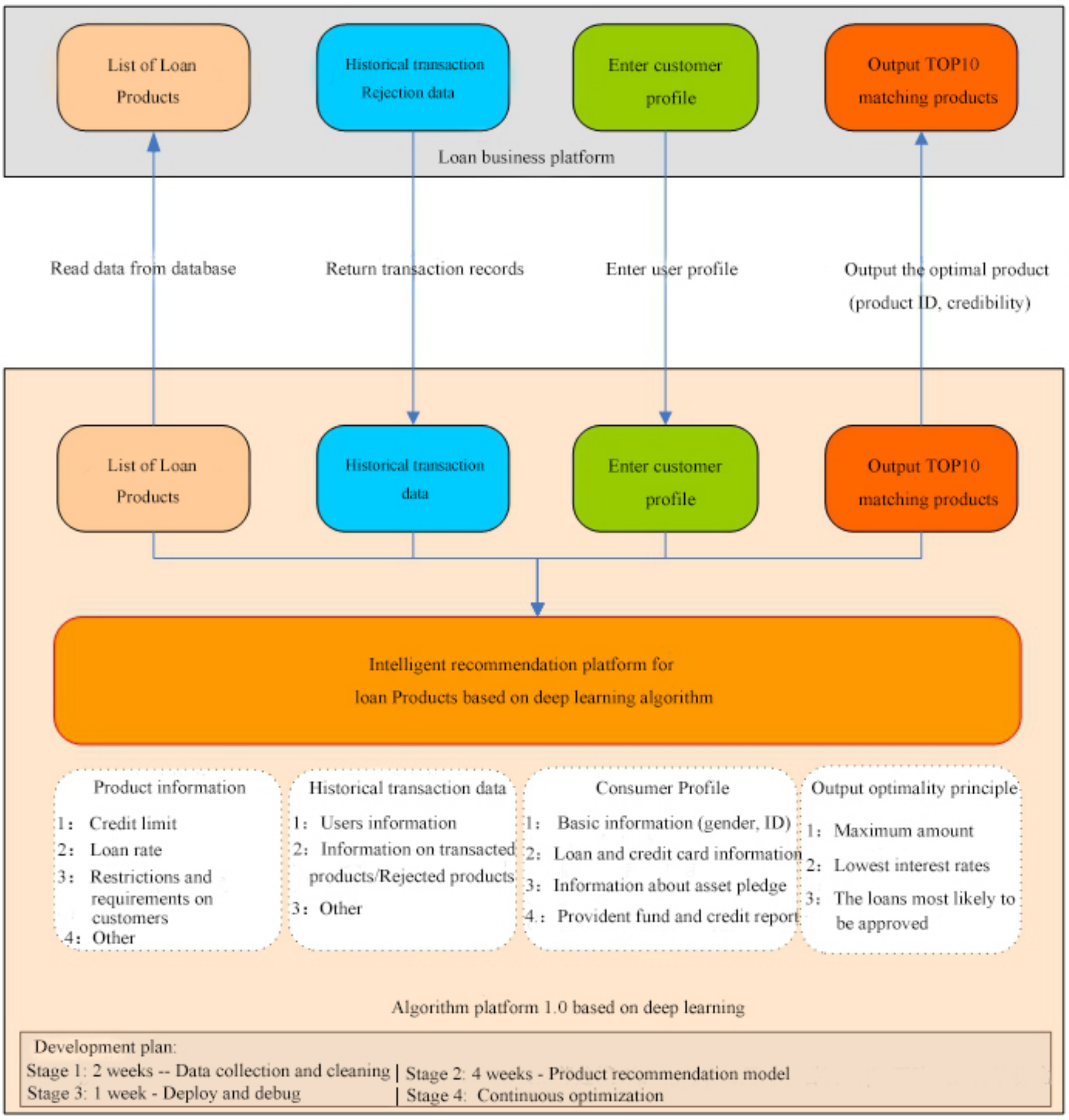
List of Loan Products
Historical transaction Rejection data
Enter customer profile
Output TOP10 matching products
Loan business platform
Read data from database
Return transaction records
Enter user profile
Output the optimal product (product ID, credibility)
List of Loan Products
Historical transaction data
Enter customer profile
Output TOP10 matching products
Intelligent recommendation platform for loan Products based on deep learning algorithm
Product information
1: Credit limit
2: Loan rate
3: Restrictions and requirements on customers
4: Other
Historical transaction data
1: Users information
2: Information on transacted products/Rejected products
3: Other
Consumer Profile
1: Basic information (gender, ID)
2: Loan and credit card information
3: Information about asset pledge
4.: Provident fund and credit report
Output optimality principle
1: Maximum amount
2: Lowest interest rates
3: The loans most likely to be approved
Algorithm platform 1.0 based on deep learning
Development plan:
Stage 1: 2 weeks -- Data collection and cleaning | Stage 2: 4 weeks - Product recommendation model
Stage 3: 1 week - Deploy and debug | Stage 4: Continuous optimization

Figure 1. System Framework

Users need to provide structured data of loan products, structured data of user portraits, loan histories and etc. Based on the data provided by users, the system uses deep learning algorithms to establish mathematical models to match users with appropriate loan products.

3 Specification

3.1 Interface

The data is packaged by JSON. The main interfaces are Login, Add , Update, Delete and Match.

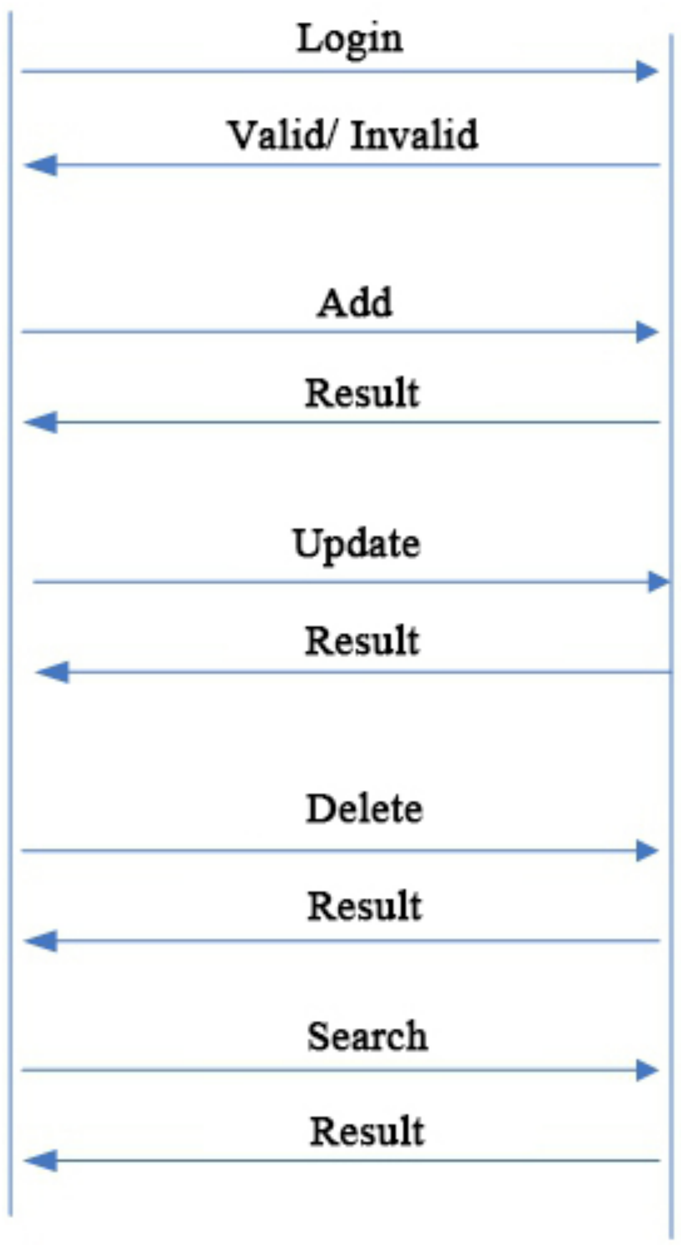

Figure 2. System interface calling process

The interfaces are classified into three categories: login verification, product maintenance, and product query.

3.1.1 Add new product

Before using the product query interface, users must add all related products to the system. The adding process is shown in the following figure.

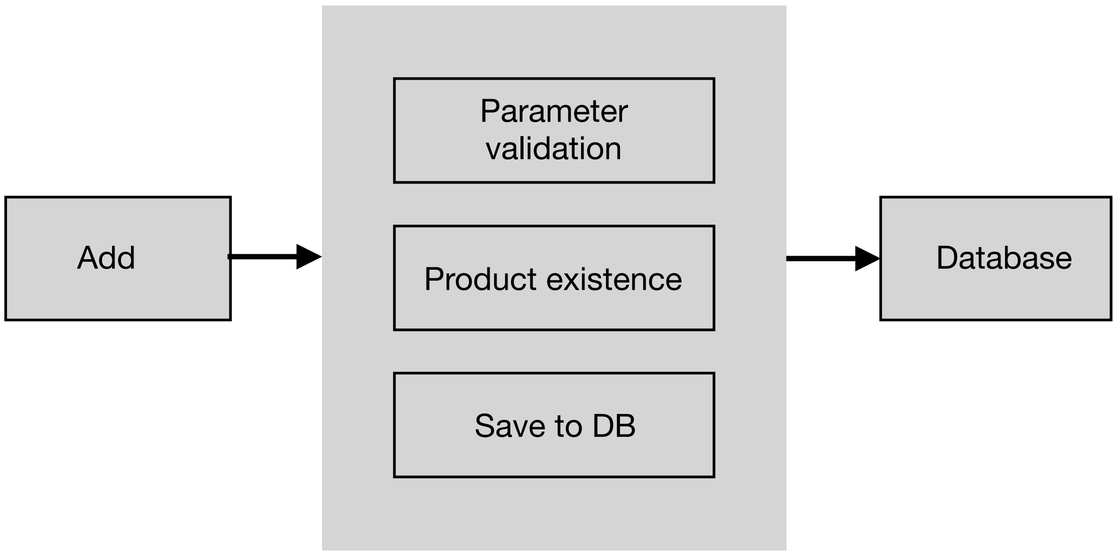

Figure 3. Add product

When adding a product, the system needs to check whether the product parameters are legal and whether the product already exists. After check, the product will be saved to the corresponding product table in the database. The interface description is described in the followings:

1) Function: When a new product is released, the algorithm is notified to Add product by calling the API;
2) Example of request address: http://HOST:PORT/api/AddProduct
3) Request method: POST

4) Request parameters

```
{
  "id" : 0 ,                  //Product ID
  "city" : "" ,                //The city where the product is located
  "bank_name" : "" ,           //Bank name
  "product_name" : "" ,         //Product name
  "loan_amount" : "" , //allowed loan amount
  " min_month_interest_rate ":0.76 // Minimum monthly interest rate
  " max_month_interest_rate ":0.98 // Maximum monthly interest rate
  "typ" : 0 ,                  //0-credit1-mortgage
  " repayment " : 0 ,            //Repayment method 0-equal principal and interest 1-first
interest and then principal 2-after interest and subsequent principal 3-equal
principal
  "repay_period" : "" ,         //Repayment period
  "min_age" : 0 ,              //Minimum age
  "max_age" : 0 ,              //Maximum age
  "is_has_gjj" : 0 ,            //Is there a provident fund requirement
  "user_type" : 0 ,             //User type 0-no 1- legal person 2- office
worker
  "has_mortgage_car_loan" : 0 ,    //Is there a mortgage car loan
  "mortgage_car_loan_month" : 0 ,    // Mortgage car repayment months
  "has_mortgage_house_loan" : 0 ,   //Is there a mortgage
  "mortgage_house_loan_month" : 0 ,  // Number of consecutive months of mortgage
repayment
  "gjj_month_payment" : 0 ,         // monthly provident fund payment
  "gjj_continuous_month" : 0 ,      // Consecutive months of provident fund
  "one_month_query" : 0 ,           // Number of queries in a month
  "two_month_query" : 0 , // Number of queries in two months
  "three_month_query" : 0 , // Number of queries in three months
  "four_month_query" : 0 , // Number of queries in four months
  "five_month_query" : 0 , // Number of queries in five months
  "six_month_query" : 0 , // Number of queries in six months
  "is_current_overdue" : 0 ,      // currently no overdue 10- is 1- No
  "current_amount" : 0 ,          // Not calculated within the current overdue
amount
  "half_year_two" : 0 ,           // Half a year without 2
  "one_year_three" : 0 ,          // one year without 3
  "two_year_four" : 0 ,           // Two years without 4
  "two_year_six" : 0 ,            // Without three consecutive years within two
years
  "credit_card_usage" : 0 ,       // credit card usage rate
  "online_loan_count" : 0 ,       // Minimum number of online loans
  "max_debt":0,              //Maximum debt (min_debt is changed to max_debt)
  "audit_intr" : "" ,            // audit instructions
  "application_materials" : "" ,   // application materials
  "add_intr" : "" ,           // Additional description
  "bank_icon" : "" , // Bank icon ;
  "status" : 0 ,              // Status: 0- normal 1- rejected
  "created_ts" : "0001-01-01T00:00:00Z" , // Created time
  "updated_ts" : "0001-01-01T00:00:00Z" , // Update time
  " Bank_type" : 0 , // type of loan: 0- Bank 1- institutions
  "Is_policy": 0 // Is there a policy requirement 0-yes 1-no
}
```

5) Request return :

```
{
"Code":200,
"Message":"OK"
}
```

3.1.2 Update product

When the parameter information of a product changes, the user needs to actively call this interface to keep the product information consistent with that product information.

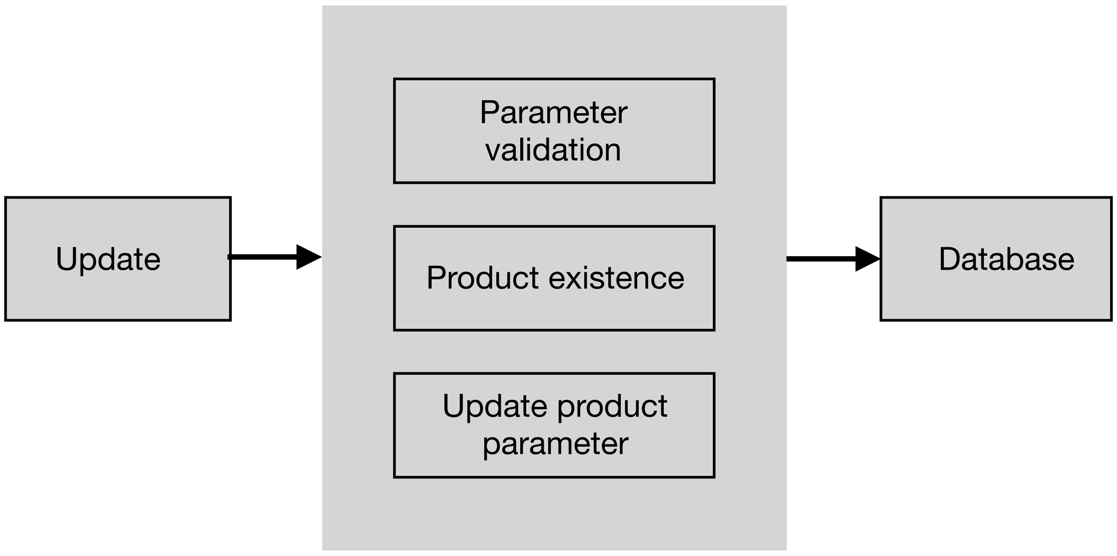

Figure 4. Update product

The process of update a product is, product parameter verification, determine whether the product already exists and update product information if it exists, then save it to the corresponding product information table in the database. The interface description is as followings:

1) Function: When there is any modification of product information , it will notify the algorithm to update product information by calling the API;
2) Example of request address: http://HOST:PORT/api/UpdateProduct
3) Request method: POST
4) Request parameters: refer to Add product chapter;
5) Request to return

```
{
"Code":200,
"Message":"OK"
}
```

3.1.3 Delete product

When a product is no longer open to users, the system needs to notify the algorithm in time to delete the product. After deletion, the product will no longer be recommended to users. The general process of deletion is shown in the following figure:

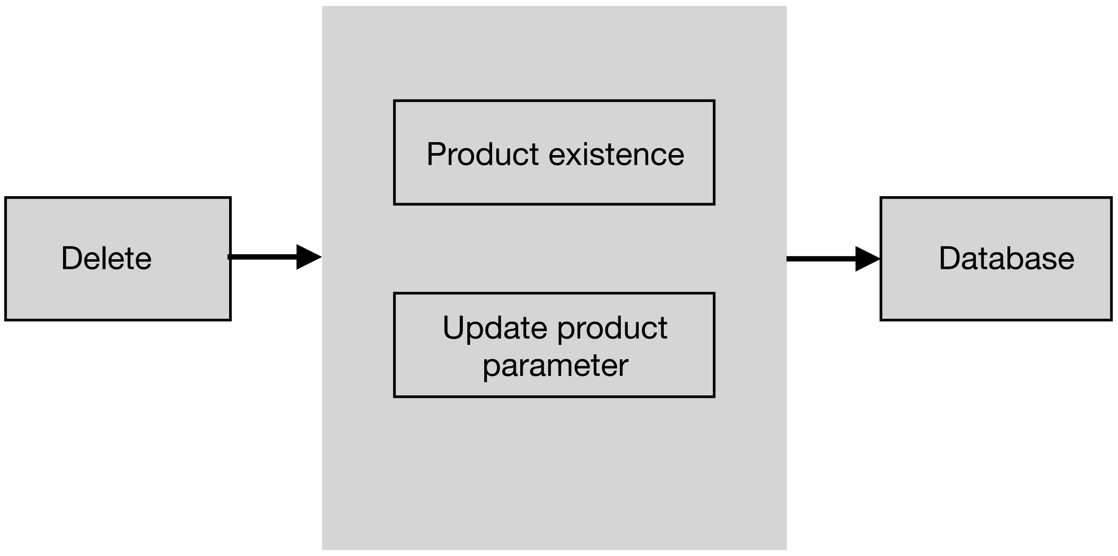

Figure 5. Delete product

1) Function: When the product information is deleted , the algorithm is notified to update the product information by calling the API;
2) Example of request address: http://HOST:PORT/api/DeleteProduct
3) Request method: POST
4) Request parameters: none
5) Request to return

```
{
"Code":200,
"Message":"OK"
}
```

3.1.4 Get matching list

Product Matching is the core part of this algorithm system. The system needs to use deep learning algorithms to profile the information input by the user. At the same time, based on the data statistics, a product model is established. Through big data analysis, it will best meet the user portrait and push a product list to the users.

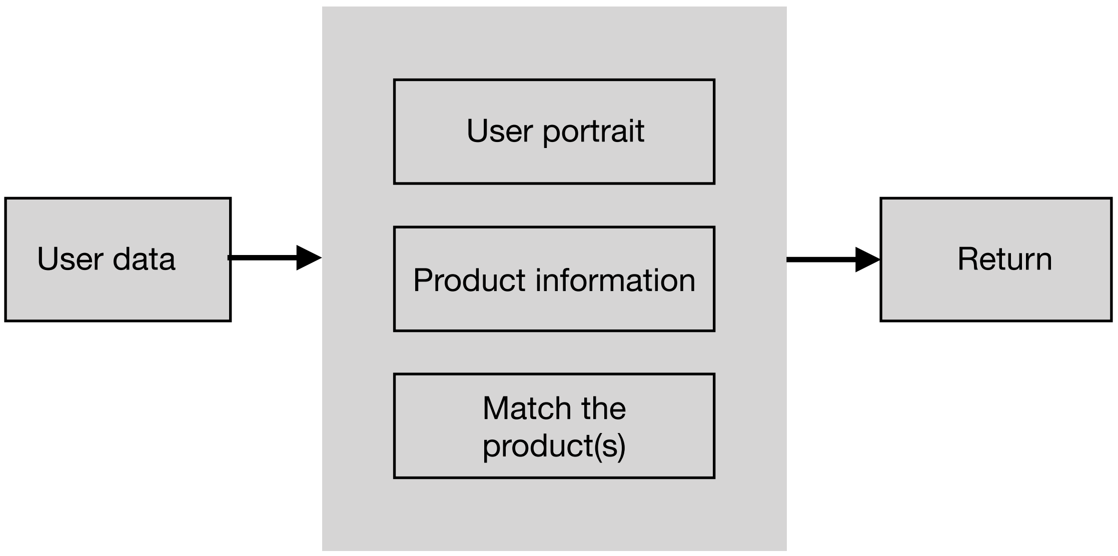

Figure 6. Product matching

1) Function: According to the user portrait, the mathematical model established by deep learning algorithm will return a product list;
2) Example of request address: http://HOST:PORT/api/optimalProduct
3) Request method: POST
4) Request parameters

Table 1. Matching Request parameters

parameter	Type	Mandatory	Description	Remarks
Age	Int	Yes	User age	Customer qualification Basic Information
Gender	int	Yes	0: unknown; 1 : male; 2 : female	
IDCard	string	Yes	identification number	
PersonType	Int	Yes	1 : Individual; 2 : Corporate legal person	
Salary	Float	Yes	Personal salary (required for individuals)	
House	Bool	Yes	True : there are rooms; False : no	
Vehicle	Bool	Yes	True: there is a car; False : no	
NeedTotalCredit	Float	Yes	The client wants the total loan amount	
SalesAmountYear	Float	Yes	Company annual turnover / 10,000 (required for legal persons)	
PayTaxesYear	Float	Yes	Annual company tax / 10,000 (required for legal persons)	
TotalCredit	Float	Yes	The total amount of credit the user has (monthly repayment)	
TotalCreditNum	Float	Yes	The total number of credits for this user (number of transactions)	
RemTotalCredit	Float	Yes	The user's remaining credit amount (total balance)	
CreditLoanInfo	Array	Yes	Credit overdue array	
TotalCreditCardAmou	float	Yes	Credit card total amount	
UsedCreditCardAmou	Float	Yes	Credit card limit used	

TotalCreditCardAmouRate	float	Yes	Utilization rate of credit card total amount	Credit card information
CreditCardInfo	Array	Yes	Credit card overdue array	
MortgageNum	int	Yes	Total number of mortgages	Mortgage information
MortgageAmo	Float	Yes	Total amount of mortgage	
RemMortgageAmo	Float	Yes	Remaining amount of mortgage	
MortgageInfo	array	Yes	Mortgage overdue array	
InstalmentsNumber	Int	Yes	Mortgage period	Mortgage loan information
FinInstalmentsNumber	Int	Yes	Number of repayment periods	
InstalmentsTotal	Float	Yes	Total mortgage	
MonthInstalmentsTotal	Float	Yes	Monthly repayment amount	
InstalmentsInfo	array	Yes	Mortgage loan overdue array	
one_month_query	int	Yes	Number of queries in a month	Credit inquiry information
two_month_query	int	Yes	Number of queries in two months	
three_month_query	int	Yes	Number of queries in three months	
four_month_query	int	Yes	Number of queries in four months	
five_month_query	int	Yes	Number of queries in five months	
six _month_query	int	Yes	Number of queries in six months	
m ax _debt	int	Yes	Maximum debt	
is_has_gjj	Bool	Yes	Provident fund	
gjj_continuous_month	int	Yes	The number of consecutive months of provident fund payment	
gjj_month_payment	int	Yes	Provident fund monthly payment amount	
is_has_netloan	Bool	thing	Is there an online loan	

netloan_number	int	Yes	Number of online loans	
is_has_bd	Bool	Yes	Is there an insurance policy	
total_credit_rate	Float	Yes	Total credit card usage rate	

Table 2. Overdue array contents

parameter	**Types**	**Mandatory**	**Description**
TotalOverdueNumber	Bool	Yes	Currently overdue (yes / no)
current_amount	Int	Yes	In case of overdue, the overdue amount
SixMonOverdueNumber	Bool	Yes	Six overdue history (Yes / No)
ThMonOverdueNumber	Bool	Yes	History overdue for three consecutive times (Yes / No)
TwoYearOverdueNumber1	Bool	Yes	Is the overdue within 2 years less than 3 (Yes / No)
TwoYearOverdueNumber2	Bool	Yes	Is the overdue within 2 years greater than or equal to 3 (Yes / No)

5) Request to return

```
{
"Code":200,
"Message":"OK"
"OptimalProduct":
{
"Total":3,
"Product":[
{
"ProductID","011",
"Confidence",0.95
},
{
"ProductID","012",
"Confidence",0.90
},
{
"ProductID","013",
"Confidence",0.85
}]
}
}
```

Return a list of 10 + products with confidence of more than 80% ;

3.1.5 Transaction order feedback

1) Function: The caller actively feeds back user's transaction information, which help optimize Credit Matching algorithm; After receiving the information, the system will inquire user table for transaction order information;
2) Example of request address : http://HOST:PORT/api/orderProduct
3) Request method: POST
4) Request parameters: none
5) Request to return

```
{
"Code":200,
"Message":"OK"
}
```

3.1.6 Reject order feedback

1) Function: The caller actively feeds back user’s reject order information, which help optimize Credit Matching algorithm; After receiving the information, the will inquire the user table for the rejected order information;
2) Example of request address: http://HOST:PORT/api/rejectProduct
3) Request method: POST
4) Request parameters: none
5) Request to return

```
{
"Code":200,
"Message":"OK"
}
```

3.2 ERROR CODES

The error code of the system starts with 10001. Each error code represents an error type. The user can customize the error code according to the situation. The system error codes currently used in the system are shown in the following table:

Table 3. Error code

No	Code	Cause of error code
1	10001	Request parameter error
2	10002	Server network disconnected
3	10003	Request too frequently