



**Faculty of Computer Science and Information Technology**

**A Hybrid Adaptive Leadership GWO Optimization with Category  
Gradient Boosting on Decision Trees Algorithm for Credit Risk Control  
Classification**

**Chen Suihai**

**Doctor of philosophy  
2024**

# A Hybrid Adaptive Leadership GWO Optimization with Category Gradient Boosting on Decision Trees Algorithm for Credit Risk Control Classification

Chen Suihai

A thesis submitted

In fulfillment of the requirements for the degree of Doctor of Philosophy

(Computer Science and Information Technology)

Faculty of Computer Science and Information Technology

UNIVERSITI MALAYSIA SARAWAK

2024

## DECLARATION

I declare that the work in this thesis was carried out in accordance with the regulations of Universiti Malaysia Sarawak. Except where due acknowledgements have been made, the work is that of the author alone. The thesis has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

*Chen Suihai*  
.....

Signature

Name: Chen Suihai

Matric No.: 21010081

Faculty of Computer Science and Information Technology

Universiti Malaysia Sarawak

Date: 12/10/2024

## **ACKNOWLEDGEMENT**

In the blink of an eye, we find ourselves on the threshold of graduation, about to venture onto divergent paths as this somber season descends upon us. Throughout the intricate journey of crafting my doctoral thesis, I have been profoundly supported by the unwavering assistance of both mentors and peers. I extend my deepest gratitude to them. First and foremost, my heartfelt appreciation goes to my supervisors, Bong Chih How and Po Chan Chiu, whose unwavering honesty, tolerance, rigor, and seriousness not only guided me through academic challenges but also enriched my personal growth. Their mentorship has been an invaluable compass navigating me through the complexities of academia and life. I am equally indebted to my wife, whose meticulous care during my recovery from surgery enabled me to swiftly regain my health and diligently pursue the completion of my thesis. Her unwavering support and selflessness have been my steadfast companions throughout this journey. Furthermore, I wish to express profound gratitude to my parents, whose unwavering support and encouragement provided me with the strength and determination to overcome every hurdle encountered along the way. Their belief in my abilities has been the driving force behind my perseverance. I am also deeply appreciative of the expertise and guidance provided by the experts and scholars whose research has inspired and influenced my own work. Their contributions have expanded my scholarly horizons and played a pivotal role in the successful completion of my thesis. Finally, I extend sincere thanks to all the reviewers and defenders who meticulously scrutinized my thesis and offered valuable insights and feedback. Your expertise and constructive criticism have undoubtedly elevated the quality and rigor of my research. As we stand on the brink of graduation, I am profoundly humbled by the collective support and encouragement I have received from all quarters.

## ABSTRACT

With the rapid development of financial economy in our country, credit business has become the main business of banks and financial companies. The quality of customer credit directly affects the business performance of financial companies. The traditional risk control model cannot identify good customers and bad customers well, so a new high precision risk control algorithm model is needed to solve this problem. CatBoost algorithm has high accuracy and takes into account the efficiency of the model, which can effectively solve such problems. However, compared with the traditional risk control algorithm (logistic regression algorithm), CatBoost algorithm also needs to have the advantages of high efficiency, low algorithm complexity and strong interpretable ability. Therefore, this study first reduces the dimension of the variable features (deleting the variable features with low correlation), and then uses two different datasets for experimental comparison and verification, which proves that the dimension reduction can improve the efficiency of the algorithm. Secondly, an improved CatBoost algorithm (EBGWO-CatBoost) was proposed, which was a combination of improved GWO algorithm (EBGWO) and CatBoost algorithm, and the optimized GWO algorithm was used to offset the defects of CatBoost algorithm in parameter tuning. The combined EBGWO-Catboost algorithm model is structured into four distinct steps. Initially, the Grey Wolf algorithm (GWO) is leveraged to exploit its advantages in identifying optimal parameters. Subsequently, the optimal parameters of the CatBoost algorithm are determined to streamline computational resource usage and reduce model complexity. However, optimizing the balance between exploration and exploitation within the GWO algorithm itself is imperative. Hence, the second step entails enhancing the GWO algorithm, referred to as the EBGWO algorithm. This enhanced version of the Grey Wolf Optimization algorithm possesses robust global search capabilities and helps alleviate some of the local

convergence issues inherent in the original GWO algorithm. It can effectively enhance the predictive accuracy and execution speed of the CatBoost algorithm model. The third step involves applying the new algorithm to the risk control model for testing and comparison, resulting in the conclusion that the model established by the EBGWO-Catboost algorithm exhibits more advantages compared to models built by other algorithms. Thirdly, this study uses SHAP framework to improve the interpretability of the new algorithm (EBGWO-CatBoost), and solves the problem of the weak interpretability of the new algorithm. Finally, a summary of the entire document is provided, along with suggestions for future work and extensions.

**Keywords:** CatBoost, GWO, EBGWO, risk control, financial industry

***Pengoptimuman GWO Kepimpinan Adaptif Hibrid dengan Peningkatan Kecerunan Kategori pada Algoritma Pokok Keputusan untuk Klasifikasi Kawalan Risiko Kredit***

**ABSTRAK**

*Dengan perkembangan pesat ekonomi kewangan di negara kita, perniagaan kredit telah menjadi perniagaan utama bagi bank dan syarikat kewangan. Kualiti kredit pelanggan secara langsung mempengaruhi prestasi perniagaan syarikat kewangan. Model kawalan risiko tradisional tidak dapat mengenal pasti pelanggan yang baik dan buruk dengan baik, jadi model algoritma kawalan risiko yang tepat tinggi diperlukan untuk menyelesaikan masalah ini. Algoritma CatBoost mempunyai ketepatan tinggi dan mengambil kira prestasi pengklasifikasian model, yang boleh menyelesaikan masalah tersebut dengan berkesan. Walau bagaimanapun, berbanding dengan algoritma kawalan risiko tradisional (algoritma regresi logistik), algoritma CatBoost juga perlu mempunyai kelebihan dari segi prestasi pengklasifikasian yang tinggi, kompleksiti algoritma yang rendah, dan keupayaan penerangan yang kuat. Oleh itu, kajian ini pertama sekali mengurangkan dimensi ciri pemboleh ubah (memadam ciri pemboleh ubah dengan korelasi rendah), dan kemudiannya menggunakan dua set data yang berbeza untuk perbandingan dan pengesahan eksperimen, yang membuktikan bahawa pengurangan dimensi boleh meningkatkan prestasi pengklasifikasian algoritma. Kedua, algoritma CatBoost yang dipertingkatkan (EBGWO-CatBoost) dicadangkan, yang merupakan gabungan algoritma GWO yang dipertingkatkan (EBGWO) dan algoritma CatBoost, dan algoritma GWO yang dioptimumkan digunakan untuk mengimbangi kekurangan algoritma CatBoost dalam penalaan parameter. Model algoritma gabungan EBGWO-Catboost disusun dalam empat langkah yang berbeza. Pertama, algoritma Serigala Kelabu (GWO) digunakan untuk memanfaatkan kelebihannya dalam mengenal pasti parameter optimum. Seterusnya, parameter optimum bagi algoritma*

*CatBoost ditentukan untuk memudahkan penggunaan sumber komputer dan mengurangi kompleksiti model. Walau bagaimanapun, mengoptimumkan keseimbangan antara eksplorasi dan eksploitasi dalam algoritma GWO itu sendiri adalah penting. Oleh itu, langkah kedua melibatkan penambahbaikan algoritma GWO, yang dirujuk sebagai algoritma EBGWO. Versi dipertingkatkan bagi algoritma Pengoptimuman Serigala Kelabu ini mempunyai keupayaan carian global yang kukuh dan membantu mengurangkan beberapa isu penumpuan tempatan yang terdapat dalam algoritma GWO yang asal. Ia boleh meningkatkan ketepatan ramalan dan kelajuan pelaksanaan model algoritma CatBoost dengan berkesan. Langkah ketiga melibatkan penggunaan algoritma baharu ini kepada model kawalan risiko untuk ujian dan perbandingan, yang menunjukkan bahawa model yang dibangunkan oleh algoritma EBGWO-Catboost mempunyai kelebihan berbanding model yang dibina dengan algoritma lain. Ketiga, kajian ini menggunakan rangka kerja SHAP untuk meningkatkan kebolehterangan algoritma baharu (EBGWO-CatBoost), dan menyelesaikan masalah kebolehterangan yang lemah bagi algoritma baharu tersebut. Akhir sekali, ringkasan keseluruhan dokumen disediakan, bersama dengan cadangan untuk kerja dan pengembangan masa depan.*

**Kata kunci:** CatBoost, GWO, EBGWO, kawalan risiko, industri kewangan

## TABLE OF CONTENTS

|                               | Page       |
|-------------------------------|------------|
| <b>DECLARATION</b>            | <b>i</b>   |
| <b>ACKNOWLEDGEMENT</b>        | <b>ii</b>  |
| <b>ABSTRACT</b>               | <b>iii</b> |
| <b><i>ABSTRAK</i></b>         | <b>v</b>   |
| <b>TABLE OF CONTENTS</b>      | <b>vii</b> |
| <b>LIST OF TABLES</b>         | <b>xii</b> |
| <b>LIST OF FIGURES</b>        | <b>xiv</b> |
| <b>LIST OF ABBREVIATIONS</b>  | <b>xvi</b> |
| <b>CHAPTER 1 INTRODUCTION</b> | <b>1</b>   |
| 1.1 Introduction              | 1          |
| 1.2 Study Background          | 2          |
| 1.3 Problem Statement         | 4          |
| 1.4 Research Questions        | 6          |
| 1.5 Objectives                | 6          |
| 1.6 Research Scope            | 7          |
| 1.7 Significance of Research  | 8          |
| 1.8 Organization of study     | 9          |

|                                                                        |           |
|------------------------------------------------------------------------|-----------|
| <b>CHAPTER 2 LITERATURE REVIEW</b>                                     | <b>10</b> |
| 2.1 Overview                                                           | 10        |
| 2.2 Research status of risk control model                              | 12        |
| 2.2.1 Compare traditional and nontraditional credit risk control model | 14        |
| 2.2.2 Research status of traditional credit risk control model         | 17        |
| 2.2.3 Research status of nontraditional credit risk control model      | 19        |
| 2.2.4 Summary status of risk control model                             | 27        |
| 2.3 CatBoost Algorithm                                                 | 29        |
| 2.4 GWO algorithm                                                      | 31        |
| 2.4.1 GWO algorithm principle and mathematical formula                 | 31        |
| 2.4.2 GWO algorithm research status                                    | 38        |
| 2.5 Research and discussion of new algorithm                           | 40        |
| 2.6 Risk control model data processing method                          | 42        |
| 2.6.1 Data processing                                                  | 42        |
| 2.6.2 WOE binning method                                               | 45        |
| 2.6.3 Screening variable                                               | 47        |
| 2.7 Chapter Summary                                                    | 48        |
| <b>CHAPTER 3 RESEARCH METHODOLOGY</b>                                  | <b>49</b> |
| 3.1 Overview                                                           | 49        |
| 3.2 Research Methodology                                               | 51        |

|                                                      |                                                        |           |
|------------------------------------------------------|--------------------------------------------------------|-----------|
| 3.2.1                                                | Problem Definition and Dataset                         | 53        |
| 3.2.2                                                | Algorithm Choices and Optimization                     | 57        |
| 3.2.3                                                | Model Validation                                       | 63        |
| 3.3                                                  | Chapter Summary                                        | 66        |
| <b>CHAPTER 4 ALGORITHM CHOICES</b>                   |                                                        | <b>67</b> |
| 4.1                                                  | Overview                                               | 67        |
| 4.2                                                  | Algorithm choices                                      | 67        |
| 4.2.1                                                | Experimental results of China_Credit dataset           | 75        |
| 4.2.2                                                | Experimental analysis of China_Credit dataset          | 78        |
| 4.2.3                                                | Experimental results of LendingClub dataset            | 79        |
| 4.2.4                                                | Experimental analysis of LendingClub dataset           | 82        |
| 4.3                                                  | Compare of CatBoost and Logistic regression Algorithms | 83        |
| 4.4                                                  | Conclusion                                             | 83        |
| 4.5                                                  | Chapter Summary                                        | 84        |
| <b>CHAPTER 5 CATBOOST CASSIFICATION OPTIMIZATION</b> |                                                        | <b>85</b> |
| 5.1                                                  | Overview                                               | 85        |
| 5.2                                                  | Features selection and algorithm test                  | 86        |
| 5.2.1                                                | Experimental results of China_Credit dataset           | 86        |
| 5.2.2                                                | Experimental analysis of China_Credit dataset          | 90        |
| 5.2.3                                                | Experimental results of LendingClub dataset            | 91        |

|                                                   |                                                                    |           |
|---------------------------------------------------|--------------------------------------------------------------------|-----------|
| 5.2.4                                             | Experimental analysis of LendingClub dataset                       | 95        |
| 5.3                                               | Conclusion                                                         | 96        |
| 5.4                                               | Chapter Summary                                                    | 97        |
| <b>CHAPTER 6 CATBOOST COMPLEXITY OPTIMIZATION</b> |                                                                    | <b>98</b> |
| 6.1                                               | Overview                                                           | 98        |
| 6.2                                               | Adaptive Leadership GWO Optimization Algorithm (EBGWO)             | 99        |
| 6.2.1                                             | EBGWO algorithm                                                    | 100       |
| 6.2.2                                             | Quantity renewal based on leadership                               | 101       |
| 6.3                                               | Pseudocode of EBGWO                                                | 104       |
| 6.4                                               | Experimental of EBGWO algorithm                                    | 105       |
| 6.4.1                                             | Experimental result of benchmarking test functions and analysis    | 106       |
| 6.4.2                                             | Experimental results and analysis of Shpere and Schwefel functions | 111       |
| 6.4.3                                             | Experimental results of China_Credit dataset                       | 117       |
| 6.4.4                                             | Experimental analysis of China_Credit dataset                      | 118       |
| 6.4.5                                             | Experimental results of LendingClub dataset                        | 118       |
| 6.4.6                                             | Experimental analysis of LendingClub dataset                       | 119       |
| 6.5                                               | Conclusion of EBGWO algorithm                                      | 120       |
| 6.6                                               | EBGWO-CatBoost algorithm                                           | 121       |
| 6.6.1                                             | Pseudocode of EBGWO-CatBoost                                       | 122       |
| 6.6.2                                             | Experimental of EBGWO-CatBoost algorithm                           | 125       |

|                                                       |                                                        |            |
|-------------------------------------------------------|--------------------------------------------------------|------------|
| 6.6.3                                                 | Experimental results of China_Credit dataset           | 125        |
| 6.6.4                                                 | Experimental discuss of China_Credit dataset           | 130        |
| 6.6.5                                                 | Experimental results of LendingClub dataset            | 132        |
| 6.6.6                                                 | Experimental discuss of LendingClub dataset            | 136        |
| 6.6.7                                                 | Experimental results and discuss of other dataset      | 137        |
| 6.7                                                   | Conclusion of EBGWO-CatBoost algorithm                 | 139        |
| 6.8                                                   | Chapter Summary                                        | 140        |
| <b>CHAPTER 7 CATBOOST EXPLAINABILITY OPTIMIZATION</b> |                                                        | <b>141</b> |
| 7.1                                                   | Overview                                               | 141        |
| 7.2                                                   | SHAP framework                                         | 141        |
| 7.3                                                   | Explainability experiment and discussion based on SHAP | 143        |
| 7.3.1                                                 | Experimental discuss of China_Credit dataset           | 143        |
| 7.3.2                                                 | Experimental discuss of LendingClub dataset            | 154        |
| 7.3.3                                                 | Conclusion                                             | 159        |
| 7.4                                                   | Chapter Summary                                        | 159        |
| <b>CHAPTER 8 CONCLUSION AND CONTRIBUTION</b>          |                                                        | <b>160</b> |
| 8.1                                                   | Conclusion                                             | 160        |
| 8.2                                                   | Research Contributions                                 | 164        |
| 8.3                                                   | Recommendation for Future Works                        | 165        |
| <b>REFERENCES</b>                                     |                                                        | <b>167</b> |
| <b>LIST OF APPENDICES</b>                             |                                                        | <b>182</b> |

## LIST OF TABLES

|                                                                            | <b>Page</b> |
|----------------------------------------------------------------------------|-------------|
| Table 2.1: Parameters description                                          | 34          |
| Table 4.1: Comparison of algorithm characteristics                         | 70          |
| Table 4.2: CatBoost important parameters                                   | 73          |
| Table 4.3: LightGBM important parameters                                   | 74          |
| Table 4.4: XGBoost important parameters                                    | 74          |
| Table 4.5: The variables of China_Credit dataset                           | 75          |
| Table 4.6: Performance indicators of different algorithm (unit: %)         | 77          |
| Table 4.7: Cases of feature variables for part of the LendingClub dataset  | 80          |
| Table 4.8: Performance indicators of different algorithm (unit: %)         | 81          |
| Table 5.1: IV and mutual information values of China_Credit dataset        | 86          |
| Table 5.2: Feature variables deletion                                      | 88          |
| Table 5.3: Performance indicators of different algorithm (unit: %)         | 89          |
| Table 5.4: IV values for some feature variables of the LendingClub dataset | 92          |
| Table 5.5: Feature variables deletion                                      | 93          |
| Table 5.6: Performance indicators of different algorithm (unit: %)         | 94          |
| Table 6.1: Expressions of unimodal functions (F1-F7)                       | 106         |
| Table 6.2: Expressions of multimodal functions (F8-F20)                    | 107         |
| Table 6.3: Test results on unimodal functions (F1-F7)                      | 109         |
| Table 6.4: Test results on multimodal functions (F8-F20)                   | 110         |
| Table 6.5: Parameter settings for shpere and schwefel functions            | 111         |
| Table 6.6: Algorithm running time                                          | 117         |
| Table 6.7: Model validation test results                                   | 118         |
| Table 6.8: Algorithm running time                                          | 119         |

|                                                           |     |
|-----------------------------------------------------------|-----|
| Table 6.9: Model validation test results                  | 119 |
| Table 6.10: Model performance evaluation Table (unit: %)  | 126 |
| Table 6.11: EBGWO-CatBoost main parameters                | 128 |
| Table 6.12: Model performance evaluation (unit: %)        | 133 |
| Table 6.13: EBGWO-CatBoost main parameters                | 135 |
| Table 6.14: Model performance evaluation (unit: %)        | 138 |
| Table 7.1: Chi square box of account balance              | 145 |
| Table 7.2: Chi square box of duration of credit (month)   | 145 |
| Table 7.3: Chi square box of length of current employment | 146 |
| Table 7.4: Examples of bad samples                        | 147 |
| Table 7.5: Good sample examples                           | 150 |
| Table 7.6: Chi square box of installment                  | 154 |
| Table 7.7: Examples of bad samples                        | 155 |
| Table 7.8: Chi square box of installment situation        | 157 |

## LIST OF FIGURES

|                                                                 | <b>Page</b> |
|-----------------------------------------------------------------|-------------|
| Figure 1.1: Chapter 1 Introduction                              | 9           |
| Figure 2.1: Principle diagram of GBDT                           | 23          |
| Figure 2.2: GWO wolf pack level                                 | 31          |
| Figure 2.3: Possible location of gray wolf surrounding prey     | 34          |
| Figure 2.4: Hunting method of wolves                            | 36          |
| Figure 2.5: Conditions for attacking prey                       | 37          |
| Figure 3.1: Research process                                    | 53          |
| Figure 3.2: Data collection card                                | 55          |
| Figure 3.3: Combined algorithm structure                        | 60          |
| Figure 3.4: Data processing procedure                           | 61          |
| Figure 3.5: Confusion matrix                                    | 64          |
| Figure 3.6: AUC structure diagram                               | 66          |
| Figure 4.1: Experimental flow structure diagram                 | 72          |
| Figure 4.2: Algorithm run time (20 variables)                   | 77          |
| Figure 4.3: Algorithm run time (110 variables)                  | 81          |
| Figure 5.1: Algorithm running time (13 variables)               | 89          |
| Figure 5.2: Algorithm running time 99 variables)                | 94          |
| Figure 6.1: EBGWO algorithm pseudocode                          | 105         |
| Figure 6.2: Tests of A in Shpere and Schwefel functions         | 112         |
| Figure 6.3: Tests of best value on Shere and Schwefel functions | 114         |
| Figure 6.4: Leadship wolves number is 5 test                    | 123         |
| Figure 6.5: Leadship wolves number is 1 test                    | 123         |
| Figure 6.6: EBGWO-CatBoost algorithm pseudocode                 | 123         |

|                                                                             |     |
|-----------------------------------------------------------------------------|-----|
| Figure 6.7: Algorithm run time (13 variables)                               | 126 |
| Figure 6.8: Model ROC Curve                                                 | 127 |
| Figure 6.9: Time profile of different algorithms running                    | 133 |
| Figure 6.10: Model ROC Curve                                                | 134 |
| Figure 7.1: Visualization of bad sample SHAP values                         | 149 |
| Figure 7.2: Visualization of SHAP values for the overall sample             | 151 |
| Figure 7.3: Account balance distribution (Classification of EBGWO-CatBoost) | 153 |
| Figure 7.4: Visualization of bad sample SHAP values                         | 155 |
| Figure 7.5: Visualization of SHAP values for the overall sample             | 156 |
| Figure 7.6: Installment distribution (Classification of EBGWO-CatBoost)     | 158 |

## LIST OF ABBREVIATIONS

|            |                                                                               |
|------------|-------------------------------------------------------------------------------|
| $P$        | Dependent variable representing the outcome to be predicted                   |
| $u$        | Random interference term in the linear regression model                       |
| $X$        | Position of the gray wolf in the search space                                 |
| $X_p$      | Position of the prey (optimal solution)                                       |
| $t$        | Current iteration number                                                      |
| $T$        | Maximum number of iterations                                                  |
| $\alpha$   | Coefficient that decreases linearly from 2 to 0                               |
| $r_1, r_2$ | Random vectors in interval $[0,1]$ used to calculate coefficients $A$ and $C$ |
| $A$        | Coefficient affecting the range of position updates                           |
| $C$        | Coefficient for influencing wolf's position relative to the prey              |
| $X_\alpha$ | Position vector of the $\alpha$ wolf (best solution)                          |
| $X_\beta$  | Position vector of the $\beta$ wolf (second-best solution)                    |
| $X_\delta$ | Position vector of the $\delta$ wolf (third-best solution)                    |
| $X_\gamma$ | Position vector of the $\gamma$ wolf (forth-best solution)                    |
| $D_\alpha$ | Distance between the current gray wolf and the $\alpha$ wolf                  |

|              |                                                              |
|--------------|--------------------------------------------------------------|
| $D_{\beta}$  | Distance between the current gray wolf and the $\beta$ wolf  |
| $D_{\delta}$ | Distance between the current gray wolf and the $\delta$ wolf |
| $D_{\gamma}$ | Distance between the current gray wolf and the $\gamma$ wolf |
| IV           | Information Value used for variable screening in risk models |
| WOE          | Weight of Evidence used for encoding variables               |
| B            | Total number of bad samples                                  |
| $B_i$        | Number of bad samples corresponding to the variable          |
| G            | Total number of good samples                                 |
| $G_i$        | Number of good samples corresponding to the variable         |
| $py_1$       | Proportion of bad samples in all bad samples                 |
| $py_0$       | Proportion of good samples in all good samples               |
| BadT         | Total number of bad samples across all variables             |
| GoodT        | Total number of good samples across all variables            |
| AUC          | Area Under Curve                                             |
| DNN          | Deep Neural Networks                                         |
| EBGWO        | Enhanced Balance Grey Wolf Optimization                      |
| GBDT         | Gradient Boosting Decision Trees                             |
| GWO          | Grey Wolf Optimization                                       |

|                     |                                        |
|---------------------|----------------------------------------|
| IV                  | Information Value                      |
| Logistic Regression | Logistic Regression                    |
| M2                  | Broad Money Supply                     |
| SHAP                | SHapley Additive exPlanations          |
| SVM                 | Support Vector Machines                |
| WOE                 | Weight of Evidence                     |
| XGBoost             | Extreme Gradient Boosting              |
| CatBoost            | Category Boosting                      |
| LightGBM            | Light Gradient Boosting Machine        |
| EBGWO-XGBoost       | EBGWO combined with XGBoost algorithm  |
| EGWO-CaTboost       | EBGWO combined with CaTboost algorithm |
| EBGWO-LightGBM      | EBGWO combined with LightGBM algorithm |

# CHAPTER 1

## INTRODUCTION

### 1.1 Introduction

With the rapid advancement of global financial markets, lending has emerged as a pivotal business for financial institutions. In recent years, propelled by the swift expansion of Internet finance, the loan business scope of traditional financial institutions has been shrinking, and traditional financial companies need to improve their own competitiveness to avoid being eliminated. To enhance competitiveness in the loan industry, financial institutions must improve risk management to swiftly and precisely identify premium customer segments. Just in this period, machine learning algorithms are gradually maturing, and their application in the financial field is becoming more and more perfect, especially in the application of financial risk control models. Further research is required to explore how this technology can be leveraged to enhance and optimize the risk control model within financial institutions, ultimately leading to a substantial enhancement in the model's overall performance in distinguishing between good and bad customers.

Currently, logistic regression is primarily used in risk control models. The algorithm processes data rapidly and is easy to use, with low complexity and good interpretability (Bücker et al., 2022). However, it has lower accuracy and limited effectiveness in distinguishing between good and bad customers. Many experts and scholars have also optimized these shortcomings and proposed many new machine learning algorithms to replace the old risk control model algorithm. Although the new algorithm has improved the accuracy of the model a lot, it has not taken into account the advantages of the traditional risk control model. Therefore, the logistic regression algorithm in traditional risk control

models has not yet been completely replaced. Consequently, this study aims to further refine the risk control model algorithm based on the contributions of these experts and scholars.

## **1.2 Study Background**

Financial companies, especially banks, have a huge impact on the national economy. Banks aim to lend money to individuals who are capable of repayment, but the global coronavirus pandemic has left negative impacts on national economies (Ajmal et al., 2021). As a result, countries began to implement quantitative easing policies, particularly those represented by the United States, which involved recklessly issuing a large amount of excess money. This action led to global inflation. Financial institutions like banks and consumer finance companies are now expanding their markets to include various target audiences. The boundaries within these groups are quite blurry, and traditional risk assessment models are unable to distinguish between them. Therefore, it is highly necessary to establish a high-differentiation risk assessment model.

The academic research on risk control models has a long history, but traditional risk control models directly reject both good and bad customers based on the pre-loan approval personnel's judgment. These personnel often face challenges such as loan fraud, black intermediaries, and grey industrial chains, which complicate the approval process. Financial companies find it difficult to handle such customers, and the key to significantly reducing overdue payments and bad debt rates lies in swiftly and precisely identifying bad customers. Enhancing the ability to identify bad customers and effectively differentiate between good and bad ones is a pressing issue that financial companies must address promptly. Therefore, the Chinese government collects data from financial companies (Hasan et al., 2022). After processing this data, experts and scholars study risk control model algorithms suitable for current financial companies.

Many studies on loan risk control and prediction exist worldwide. Traditional methods mainly combine financial indicators, personal credit data, loan willingness, and other aspects. However, low accuracy arises from using the logistic regression algorithm in establishing risk control models. Abundant studies on credit evaluation are primarily based on financial indicators and expert opinion formation models (Fernandes et al., 2016). Verification has led to the proposal of a credit evaluation model based on logistic regression, which has become one of the mainstream methods for user credit evaluation. Subsequently, Liang et al. (2023) analyzed the causes behind the failures of listed companies in Shanghai and Shenzhen. They highlighted that the principal component discriminant model has significant application value in risk monitoring and credit evaluation. Despite this, the low accuracy of traditional regression models makes it challenging to achieve high prediction effectiveness. Consequently, many experts use different combinations of algorithms to study risk control models. For instance, experts (Szymura, 2022; Blanco et al., 2013) have adopted a blend of multiple machine learning algorithms to enhance their models' performance. The combined classification algorithm with machine learning forest has proven to possess higher accuracy and stability in loan risk prediction algorithms. Ruan et al. (2021) demonstrated that the accuracy of an integrated machine learning model surpasses that of a single model by comparing six credit risk prediction methods for SMEs. Additionally, this model exhibits high accuracy and recall rates, thus playing a crucial role in the internet financial risk control system.

These results prove that the accuracy of combined machine learning algorithm models is higher than that of single algorithm models. However, combined models still have some flaws. Although the combined algorithm model has high precision, it suffers from slow classification performance. Adjusting and calculating model parameters require significant

time and computing memory, which increases model complexity and negatively impacts its effectiveness. These literature studies indicate that the composite algorithm retains only some advantages found in the traditional logistic regression model. Therefore, further research is required to address and resolve these issues.

### **1.3 Problem Statement**

To improve their ability to compete in the market, financial companies need to swiftly and precisely distinguish between good and bad customers. This capability can increase their profits. In fact, a large group of customers falls between being good or bad, and traditional risk assessment methods cannot effectively differentiate them. While logistic regression in traditional risk control models is fast, simple, and easy to interpret (Stoltzfus, 2011), it cannot effectively distinguish between good and bad customers. Therefore, enhancing the accuracy of a model that is fast, easy to use (low complexity), and provides good explanation is crucial for improving traditional risk control algorithms. Logistic regression is suitable for solving linear problems, but it has shortcomings, such as low accuracy and an inability to handle nonlinear data. Kaastra and Boyd (1996) proposed that an ideal risk control model should quickly distinguish between good customers (good credit) and bad customers (bad credit) while maintaining good accuracy, strong stability, strong explainability, good universality, and low consumption of computing resources (enterprise cost).

Many scholars (Ma et al., 2018) currently use XGBoost and LightGBM algorithms for data testing. The results from these algorithms demonstrate significant improvement compared to those achieved using the traditional logistic regression algorithm. Additionally, researchers (Chang et al., 2018) suggest using XGBoost as an alternative to logistic regression. Some scholars (Ma et al., 2018) have also combined the logistic regression algorithm with the XGBoost algorithm and achieved notable results. However, XGBoost has

its drawbacks, including overfitting, difficulty in parameter tuning, and slow learning speed. On the other hand, LightGBM offers high speed and less resource consumption but has lower accuracy compared to XGBoost (Liang et al., 2020). The new CatBoost algorithm addresses many of the shortcomings of XGBoost and LightGBM (Bentéjac et al., 2021). That is, 1. Logistic regression algorithm, Advantages: high training speed, stability, and good interpretability. Disadvantages: accuracy is not high. 2. LightGBM: high speed, less resource consumption, but lower accuracy than XGboost. 3. XGBoost algorithm, advantages: high accuracy. Disadvantages: slow. 4. CatBoost: high speed and high accuracy, CatBoost algorithm which is compatible with the performance of the three algorithms.

The main advantages of the logistic regression algorithm are its speed, simplicity (ease of use), and good interpretability. Financial institutions, such as banks, need to quickly identify good or bad customers, and logistic regression is very simple to use and can explain why a customer is classified as good or bad. However, the accuracy in identifying customers who fall between good and bad is very low. Although current new algorithms address the low accuracy issue of traditional logistic regression, they cannot fully match the main advantages of logistic regression.

From the research of experts and scholars, it is found that although these new algorithms can solve most of the defects of the traditional risk control model, they do not have the main advantages of the original algorithm (logistic regression algorithm), so they cannot completely replace logistic regression algorithm in risk control model.

Assume that there is an algorithm capable of improving the accuracy of the risk control model. If this algorithm undergoes further optimization and eventually incorporates the main advantages of logistic regression (such as fast computation speed, ease of use, and strong

interpretability), then this algorithm could replace the original algorithm (logistic regression). Therefore, finding such an algorithm is key to solving the problem.

#### **1.4 Research Questions**

The main research problems studied in this thesis are as follows:

1. How to improve the accuracy of the existing risk control model, the ability to identify good customers and bad customers, and at the same time, it has a high efficiency of model operation.
2. How to simplify the complexity of model algorithm while improving the accuracy of risk control model.
3. How to improve the explainability of the risk control model.

#### **1.5 Objectives**

Overall research aim:

To enhance the effectiveness of risk control models by developing and integrating advanced algorithms and techniques that improve accuracy, parameter optimization, and interpretability, thereby addressing the limitations of traditional models and providing a more robust solution for distinguishing between good and bad customers. It improved model accuracy by about 15% and classification performance (running speed) by about 20%.

Objectives:

- a. Explore and Improve Accuracy: 1. Investigate various advanced new algorithms, including but not limited to the CatBoost algorithm, to improving the accuracy of traditional risk control models. The new algorithm improves the accuracy of risk control models by about 10%. 2. Evaluate and compare the performance of these algorithms, in terms of their ability to accurately

identify good and bad customers, and assess their impact on model accuracy.

3. Examine potential methods for dimensionality reduction and variable selection to enhance new algorithm classification performance. The classification performance of the new algorithm is improved by about 20%.

- b. Optimize Parameters and Simplify Complexity: 1. Explore different optimization techniques, such as the Grey Wolf Optimization (GWO) algorithm, to identify optimal parameters for advanced algorithms like CatBoost. 2. Investigate the limitations of existing optimization techniques and explore new approaches, such as an enhanced GWO algorithm, to address issues like local optima and balance exploration with exploitation. 3. Assess the impact of these optimization techniques on simplifying the complexity of the model algorithm. The accuracy of the model algorithm is further improved by about 5%.
- c. Enhance Interpretability: Use SHAP (SHapley Additive exPlanations), to improve the interpretability of risk control models. Investigate how different algorithms and their optimizations affect the model's explainability.

## **1.6 Research Scope**

This study explores a new algorithm that integrates the optimized Grey Wolf Optimization algorithm with the CatBoost algorithm. The dataset used for the experiment, China\_Credit (refer to Chapter 3, Section 3.2.1), is provided by the Chinese government from financial companies, and is processed for use by commercial banks, experts, and scholars. To assess the model's performance and versatility, the study also employs the publicly available Lending Club dataset for testing and comparison.

## 1.7 Significance of Research

This study aims to enhance the model's accuracy using the CatBoost algorithm. However, the CatBoost algorithm has drawbacks, as it requires substantial memory and time to process categorical feature variables, and it needs to manually adjust the parameters to obtain high accuracy. Therefore, this study tried to use IV (Information value, see section 2.5.3) and mutual information to calculate the importance of feature variables in the dataset, and then carried out variable screening to delete the feature variable with the highest correlation degree, a process called dimensionality reduction. This approach aims to diminish the time and computational resources required by the CatBoost algorithm in processing feature variables. Simultaneously, CatBoost algorithm is optimized, leveraging the strengths of other algorithms to compensate for its shortcomings. Grey Wolf Algorithm (GWO) has strong advantages in finding the optimal parameters. Therefore, this study leverages the Grey Wolf Optimization (GWO) algorithm to automatically determine the optimal parameters of the CatBoost algorithm, eliminating the need for manual parameter adjustments. This approach significantly enhances algorithm efficiency and reduces complexity. However, further optimization is required to refine the balance between the exploration and exploitation capabilities of the GWO algorithm, and this study attempts to improve the GWO algorithm, hereby referred to as the EBGWO algorithm. The Enhanced Grey Wolf Optimization (EBGWO) algorithm represents an enhancement over the original Grey Wolf Optimization algorithm, exhibiting superior global search capabilities. It addresses the local convergence issue inherent in the Grey Wolf Optimization algorithm to a certain extent, resulting in enhanced prediction accuracy and faster runtime for the CatBoost algorithm model. Finally, SHAP framework is used to further enhance the explainability of the new algorithm.

## 1.8 Organization of study

This research thesis is structured into eight chapters, as depicted in Figure 1.1. The first chapter provides a brief overview of background of risk control model issue, elucidates research question, purpose, and significance of this thesis. The second chapter is dedicated to a comprehensive literature review. This chapter first introduces the research process and research status of the risk control model in detail. Secondly, it elaborates on the latest research status of the risk control model algorithm and finally introduces the processing method of the risk control model dataset. Chapter 3 introduces the research methods, which are described in detail. Chapter 4 introduces selection of algorithm. Chapter 5 introduces the efficiency optimization of the CatBoost algorithm. Chapter 6 delves into the intricacies of optimizing the complexity of the CatBoost algorithm, which is substantiated through experiments conducted with two distinct datasets. Chapter 7 describes the use of SHAP framework to enhance the interpretability of new algorithms. Chapter 8 summarizes the thesis and Outlines the future research direction.

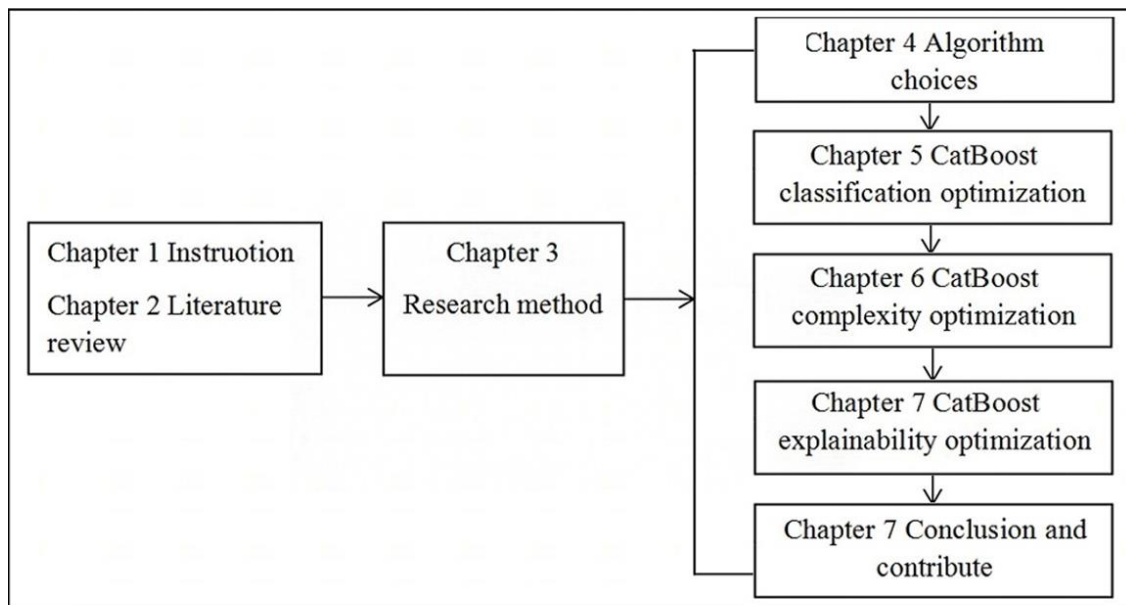


**Figure 1.1: Chapter 1 Introduction**

## **CHAPTER 2**

### **LITERATURE REVIEW**

#### **2.1 Overview**

This chapter discusses the evolution of credit risk assessment methods, the introduction of advanced algorithms for model optimization, and the importance of rigorous data processing and feature engineering. By integrating modern techniques and optimizing data handling, the proposed approach aims to enhance the performance and accuracy of credit risk models. The main contents are summarized as follows:

##### **2.1.1. Traditional in Modern Credit Risk Assessment Methods**

Credit risk assessment has evolved from traditional methods such as credit scoring models and logistic regression to advanced techniques leveraging machine learning (ML) and deep learning (DL). Traditional models, while foundational, are limited by their assumptions, data constraints, and rigidity. Traditional models often assume linear relationships and may struggle with high-dimensional or non-linear data. Modern ML and DL approaches, including Random Forests, Support Vector Machines (SVMs), Gradient Boosting Machines (GBMs), and Neural Networks, address these limitations by capturing complex, non-linear relationships and improving accuracy and adaptability. However, these advanced methods face challenges like interpretability and increased computational complexity.

##### **2.1.2 Advances in Risk Control Algorithms**

Recent advancements introduce sophisticated algorithms like CatBoost and optimization techniques such as the Grey Wolf Optimizer (GWO) to bridge the gaps in traditional methods. 1. CatBoost: Developed by Yandex, CatBoost excels in handling categorical features and reducing overfitting. CatBoost transforms categorical data into numerical values through sophisticated encoding and binarization techniques, which enhances its performance in feature processing. 2. Grey Wolf Optimizer (GWO): Inspired by the hunting behavior of gray wolves, GWO is known for its simplicity, minimal parameter adjustment, and effective convergence. GWO balances exploration and exploitation in optimization tasks. Variants of GWO, like EGWO and PGWO, enhance GWO's performance by incorporating additional strategies for better parameter optimization and local search capabilities.

#### 2.1.3. Optimizing the Risk Control Model

To improve the efficiency and accuracy of risk control models, a hybrid approach combining CatBoost with an enhanced GWO algorithm (EB-GWO) is proposed. The EB-GWO algorithm integrates adaptive leadership into GWO to refine its search capabilities, thus optimizing CatBoost parameters more effectively. This combination aims to enhance model convergence, reduce parameter complexity, and streamline operations.

#### 2.1.4. Data Processing Techniques

Effective data processing is crucial for model performance. Key steps include: 1. Handling Missing Values: Using methods like single imputation and multiple imputations addresses gaps in data. 2. Outlier Processing: Identifying and managing outliers prevents skewed results. 3. Defining Good and Bad Customers: Using criteria like M1+, M2+, and M3+ classifies customers based on overdue periods. 4. Sample Size and Window Period:

Determining appropriate sample sizes and defining observation and performance periods for data collection are essential.

#### 2.1.5. Feature Engineering and Variable Selection

WOE Binning Method: The Weight of Evidence (WOE) method transforms categorical variables into a continuous format suitable for logistic regression, which helps manage outliers and creates monotonic relationships. 1. Information Value (IV): Information Value (IV) assesses the predictive power of variables and guides feature selection by identifying variables with strong associations to the target outcome.

## 2.2 Research status of risk control model

The research status of risk control model is as follows: 1. Introduction to Credit Risk Assessment (Edunjobi & Odejide, 2024). Credit risk assessment is a crucial component of financial risk management, involving the evaluation of the likelihood that a borrower will default on a loan. Accurate assessment is vital for lenders to minimize losses and optimize their portfolios. The process typically involves both qualitative and quantitative methods to predict and manage risk. 2. Traditional Credit Risk Assessment Techniques (Toh, 2023). Credit Scoring Models: Traditional methods such as FICO scores or credit ratings have been widely used. These models are often based on historical credit performance and financial metrics like debt-to-income ratios and payment history. They provide a baseline for assessing creditworthiness but may not capture complex, non-linear relationships in the data. Statistical Methods: Techniques like logistic regression and discriminant analysis have been used to model default probabilities (Abdulhafedh, 2022). These methods are well-understood and relatively straightforward but may struggle with high-dimensional or non-linear data. 3. Limitations of Traditional Methods (Sharma et al., 2024). Assumptions and

Simplifications: Traditional methods often rely on linear assumptions and may not account for all factors influencing credit risk. Data Constraints: These models may be limited by the availability and quality of historical data. Changes in economic conditions or borrower behavior can also impact their effectiveness. Model Rigidity: Standard models may not adapt well to emerging patterns or new types of risk, potentially leading to inaccurate predictions.

4. Recent Advances in Credit Risk Assessment (Edunjobi & Odejide, 2024). Machine Learning Approaches: Recent advancements have introduced machine learning techniques that can handle non-linearity and high-dimensional data more effectively. Techniques such as Random Forests, Support Vector Machines, and Gradient Boosting Machines have shown improved performance over traditional methods. Deep Learning: Neural networks and other deep learning approaches have been applied to credit risk modeling, offering potential improvements in predictive accuracy through complex feature interactions. Ensemble Methods: Combining multiple models to improve prediction robustness and accuracy has gained traction. Techniques like stacking and boosting have been effective in refining credit risk assessments. 5. Specific Challenges and Limitations (Addy et al., 2024). Interpretability: While advanced models like deep learning offer high accuracy, they often lack transparency, making it difficult to understand and interpret their predictions. Overfitting: Complex models risk overfitting to historical data, which may not generalize well to new or unseen situations. Data Privacy and Ethics: Handling sensitive financial data requires stringent measures to ensure privacy and ethical use.

The credit risk assessment method combines mathematical statistics, financial engineering, and computer technology to determine a borrower's default risk. By building a model, financial companies can distinguish between "good customers" and "bad customers," or "normal customers" and "abnormal customers." Typical statistical models for supervised

learning classification algorithms include univariate models, multivariate discriminant models, linear models, logistic regression models, decision tree models, support vector machines, cluster analysis, K-nearest neighbor discriminant models, integrated algorithms, neural network technology, and genetic algorithms. Additionally, there are non-statistical methods such as combinatorial models, geometric evaluation theory, projection pursuit, and discriminant methods (Das et al., 2021).

### **2.2.1 Compare traditional and nontraditional credit risk control model**

Most traditional risk control models adopt a logistic regression algorithm (Bolton et al., 2010). The logistic regression model operates on the principles of generalized linear modeling. It transforms the correlation between predictor variables and the target variable into probabilities. The primary advantages of the logistic regression model include: 1. Simplicity and ease of explanation. The output value ranges between 0 and 1, which helps in explaining how each input variable influences the model's output. 2. High computational classification performance. The model allows for fast training and prediction, even with large-scale datasets. 3. Capability to handle multi-category problems. The logistic regression model can address multi-class problems directly without requiring one-to-many or many-to-many transformations. However, the logistic regression model also has some shortcomings: 1. Insensitivity to nonlinear relationships. As a linear model, it struggles to capture nonlinear relationships effectively. 2. Large deviations may occur due to its linear nature. 3. Underfitting issues arise because of the model's simplicity. This simplicity, which resembles a linear model, hampers the accurate depiction of the underlying data distribution, leading to reduced accuracy.

Non-traditional risk control model is modeled using advanced algorithms, such as the neural network algorithm (see section 2.2.3.4), SVM algorithm (support vector machine

algorithm, see section 2.2.3.5), boosting (see section 2.2.3.6) algorithm, etc. The classification performance of credit risk control models can vary significantly depending on the algorithm used. Here's a general overview of how these algorithms perform in the context of credit risk assessment, based on recent studies and benchmarks:

1. Kernel Logistic Regression: Kernel logistic regression extends traditional logistic regression by incorporating kernel functions to handle nonlinear relationships. This method can improve performance compared to standard logistic regression but is still generally less powerful than some other advanced methods. Classification Performance: Kernel logistic regression can achieve performance metrics similar to or slightly better than traditional logistic regression, particularly in cases where nonlinearities are moderate. However, it may not match the performance of more advanced models like deep neural networks or boosting methods.
2. Interpretable Deep Learning-based Credit Assessment Approach (DNN). Deep Neural Networks (DNNs) are capable of modeling complex, nonlinear relationships and can capture intricate patterns in credit data. Classification Performance: DNNs often deliver high classification performance, with accuracy and AUC-ROC scores typically surpassing those of traditional and kernel-based methods. However, achieving high performance with DNNs depends on careful tuning of hyperparameters and sufficient training data. They are known for high predictive accuracy but can suffer from interpretability issues, even when efforts are made to enhance interpretability.
3. Support Vector Machine (SVM). SVMs, particularly with the use of kernel functions, are effective at handling high-dimensional and nonlinear data. Classification Performance: SVMs generally offer strong performance, especially in binary classification problems. They often achieve high accuracy and AUC-ROC scores. SVMs can be particularly effective when the data has a clear margin of separation. However, performance can degrade with very large datasets or if the choice of the kernel is not optimal.

4. Boosting: Boosting algorithms (e.g., Gradient Boosting, XGBoost) create a strong predictive model by combining multiple weak learners (typically decision trees) in an iterative manner. Classification Performance: Boosting methods usually provide excellent classification performance, often achieving high accuracy and AUC-ROC scores. They are very effective in dealing with a variety of data patterns and can handle complex relationships and interactions between variables. They also provide good robustness against overfitting when tuned correctly.

The advantages of these algorithms are: that they can solve nonlinear problems, and the accuracy is relatively high. Disadvantages: the algorithm structure is complex, and the explainability is weak (Yang et al., 2022).

1. Complexity: Increased Computational Resources: Complex algorithms often require more computational power and memory to process data. Steeper Learning Curve: Developers and data scientists might face a steeper learning curve when working with complex algorithms, which can slow down the adoption and integration process within teams or organizations. Implementation Challenges: Complexity can lead to challenges in implementation, such as difficulties in tuning hyperparameters or integrating the algorithm with existing systems.
2. Explainability: Lack of Transparency: Algorithms with weak explainability often operate as "black boxes", making it difficult to understand how they arrive at specific decisions or predictions. Difficulty in Diagnosing Errors: Without clear explanations, it can be challenging to diagnose and correct errors or biases in the algorithm.

These algorithms can solve most of the shortcomings of logistic regression algorithms, but they can not fully inherit the advantages of logistic regression and only have some advantages of logistic regression (Martín-Baos & Garcia-Rodenas, 2021). Non-traditional models, such as neural networks or Support Vector Machines (SVMs), excel in handling

complex, nonlinear relationships. Logistic regression, however, struggles with these due to its linear nature. Logistic regression uses the logistic function,  $P(Y=1|X) = 1/(1+e^{-(\beta_0 + \beta_1*X_1 + \beta_2*X_2 + \dots + \beta_k*X_k)})$ . This function limits logistic regression to linear decision boundaries. Neural networks use activation functions like ReLU (Rectified Linear Unit) or sigmoid to introduce non-linearity. SVMs use kernel functions to map data into higher dimensions, which allows them to model complex patterns.

However, non-traditional models typically involve more intricate algorithms, which can lead to decreased interpretability. Logistic regression provides straightforward coefficients,  $\beta$ , indicating the effect of each predictor on the probability, whereas complex models often result in less transparent decision-making processes, making it harder to directly interpret their results. Thus, while non-traditional models address logistic regression's limitations, they do not fully inherit its simplicity and ease of explanation. So non-traditional models can't completely replace traditional models.

### **2.2.2 Research status of traditional credit risk control model**

The logistic regression model is highly prevalent in banks and other financial institutions, serving as one of the most widely adopted methods in the industry (Bekhet & Eletter, 2014). Certainly, there are different algorithms like multiple linear regression and discriminant analysis algorithms, among others (Li et al., 2023).

When applying the multiple linear regression method to the credit scoring model, it can be described as follow  $P = \beta_0 + \beta_1*X_1 + \beta_2*X_2 + \dots + \beta_k*X_k + u$ , where  $u$  is random interference term.  $P$  is the dependent variable, which represents the outcome that we want to predict.  $X_1, X_2, \dots, X_k$  is an independent variable, these factors represent the variables that influence the dependent variable within the logistic regression model.  $(\beta_0, \beta_1 + \beta_2 \dots + \beta_k)$  is a regression

coefficient, it is a plural. Its relationship signifies the association between the dependent variable and the independent variable within the logistic regression model. The application of linear regression analysis to personal credit rating has obvious defects. In this model, the value of the dependent variable can range from negative infinity to positive infinity, but the range of the default probability of customer lies within the interval of  $[0,1]$ . If the left side of the equation is transformed into a function of  $P$ , which can take in any value, then the model makes more sense.

The discriminant analysis algorithm (Dinca & Bociu, 2015) is used extensively for classifying instances into different categories based on their characteristics. This method first establishes discriminant criteria and functions from instances of known categories. To classify a new instance, each feature variable of the unknown instance is substituted into the discriminant functions for each category, and the resulting discriminant values are compared. Discriminant analysis requires several strict assumptions: it assumes no multicollinearity among feature variables and expects a multivariate normal distribution among variables, with equal covariance matrices for each group. When these assumptions are not met, the accuracy of discriminant analysis can be significantly compromised. Despite this, discriminant analysis is still widely used in personal credit scoring due to its robustness, though it may lack in accuracy (Khemakhem & Boujelbene, 2015).

Berkson (1944) proposed the Logistic Regression algorithm, which has several advantages. The algorithm trains rapidly, and its computational load primarily depends on the number of features involved. The analysis is straightforward and easy to understand, providing high model explainability. The model's interpretability is notable, as it allows the influence of various features on the final outcome to be discerned through feature weights. This approach is highly suitable for binary classification tasks because it does not require

scaling of input features and places minimal demands on computational resources, such as memory usage. Devika et al. (2022) applied logistic regression algorithms to credit risk control and obtained favorable results. The results are that the algorithm is fast and the accuracy is improved to a certain extent. This method, using logistic regression to establish the risk control model, offers advantages such as simplicity, ease of use, robust explanatory capabilities, and swift execution speed. While it provides quick and straightforward results with some explanatory value, its accuracy is relatively low.

However, logistic regression struggles with nonlinear problems due to its inherently linear decision boundaries. It also faces challenges in handling data imbalances, resulting in lower accuracy. For credit risk models, this might be the performance of a basic logistic regression or a credit scoring model. Benchmarked citation accuracy can vary widely, but for many traditional models, accuracy rates often range between 60% and 80%, depending on the quality of data and the specific context (Brentnall et al., 2018). To assess the effectiveness of your new model, compare its accuracy against this baseline. Therefore, the current research can not completely solve the problem of financial companies. In other words, the risk control model of financial companies needs to be solved with high accuracy, and at the same time, the model has the advantages of simple use, strong interpretation ability and fast execution speed.

### **2.2.3 Research status of nontraditional credit risk control model**

Identifying Pitfalls in Existing Credit Risk Control Techniques (Bhattacharya et al., 2023). 1. Limitations of Traditional Models. Traditional credit risk models, such as logistic regression and credit scoring systems, often rely on simplistic assumptions. These models may not capture complex, non-linear relationships between features and default risk. For example, logistic regression assumes a linear relationship between predictor variables and

the log-odds of the outcome, which can be restrictive when dealing with real-world data that exhibits more complex patterns. 2. Accuracy Issues in Dynamic Environments. Existing methods sometimes struggle to maintain accuracy in rapidly changing economic conditions. Traditional models might not adapt well to new types of risk or shifts in borrower behavior. For instance, a model trained on historical data may fail to predict defaults accurately during economic downturns or in the face of new financial products. 3. Data Quality and Availability. Many credit risk models are limited by the quality and completeness of the data used. Issues such as missing values, data bias, and outdated information can undermine the reliability of predictions. For example, models based on historical credit performance might not fully account for recent trends or emerging risk factors. 4. Challenges with Interpretability. As models become more complex, such as with the introduction of machine learning algorithms, interpreting their predictions can become difficult. This lack of transparency can be a significant drawback, particularly in regulatory environments where understanding the rationale behind credit decisions is crucial.

The research direction of this study is as follows: 1. Research and explore the machine algorithm with high accuracy to solve nonlinear problems. 2. Explore and study risk control models using simple methods. A combination of different algorithms can be used for experimental exploration. 3. Use IV and mutual trust information to reduce the dimension of the model and improve the running time of the model algorithm. 4. The introduction of SHAP framework can improve the accuracy of algorithm model.

Many studies have explored the shortcomings of traditional risk control models. For instance, Kamaloo (2014) addressed these shortcomings by using fuzzy learning methods, while Satsiou (2005) employed genetic algorithms in personal credit risk assessment. Zhang et al. (2009) developed a fusion evaluation model based on DS evidence theory. This model

combined the Support Vector Machine (SVM) and neural network approaches. By leveraging the strengths of both algorithms, their unified modeling approach significantly enhanced accuracy.

From 2022 to 2024, research on CatBoost focused on several key areas. The model training efficiency was further optimized, methods for handling categorical features were improved, and advanced hyperparameter tuning techniques were introduced to enhance performance. Additionally, there were efforts to speed up model training on large-scale data, improve handling of extreme class imbalance, and enhance support for sequential and graph data. These advancements aim to improve model applicability and efficiency, with specific optimizations for financial and medical data.

#### 2.2.3.1. Artificial neural networks

Artificial Neural Networks (ANNs) are a type of technology that emerged in the 1990s, simulating biological neural networks. These systems consist of numerous interconnected neurons that mimic the structure and functionality of the human brain. The neurons simulate how brain nerves process information and perform parallel processing and nonlinear transformations. Through training on sample data, neural networks exhibit memory and recognition capabilities similar to those of the human brain. They serve as powerful tools for addressing complex nonlinear problems that traditional statistical classification methods may struggle with. Alongside logistic regression, ANNs are among the most widely utilized models in personal credit scoring. Asha et al. (2021) used ANNs to effectively distinguish between fraudulent and non-fraudulent transactions. Similarly, Esenogho et al. (2022) conducted experiments using publicly available real-world credit card transaction datasets and found that ANNs effectively differentiated between fraudulent and non-fraudulent

transactions. The main advantage of ANNs is their high classification accuracy, while the main disadvantages are a lack of interpretability and stability.

#### 2.2.3.2. Support vector machine

The theory of Support Vector Machines (SVMs) emerged in the mid-1990s, focusing on machine learning principles in small-sample scenarios. SVMs derive from the concept of optimal classification hyperplanes in linearly separable cases. Using SVMs for credit risk assessment offers advantages such as high classification accuracy and suitability for small sample sizes. However, SVMs are sensitive to training parameters and can lack robustness. Tiwari et al. (2021) combined SVMs with other nonlinear algorithms, such as logistic regression, to address the challenge of handling nonlinear data, though the algorithm's accuracy remains limited. Xie et al. (2021) used the SVM algorithm to experiment with nonlinear credit loan data, finding superior performance compared to the logistic regression algorithm traditionally used in risk control models.

Tiwari et al. (2021) combined SVM (Support Vector Machine) with other nonlinear algorithms, such as logistic regression, to effectively address the limitation of handling nonlinear data. However, the accuracy of the algorithm is not high. Xie et al. (2021) used the SVM (Support Vector Machine) algorithm was utilized to conduct experiments on nonlinear credit loan data, and the results indicated superior performance when compared to the logistic regression algorithm traditionally used in risk control models.

#### 2.2.3.3. Boosting algorithm

With time, the research on boosting algorithms is increasing gradually, because this kind of algorithm has higher accuracy (Gamini et al., 2021). Boosting algorithm mainly includes the Adaboost (Adaptive Boosting) algorithm and the mainstream GBDT (Gradient

Boosting Decision Tree) algorithm framework. Building upon the framework of Gradient Boosting Decision Trees (GBDT), this study primarily utilizes algorithms such as XGBoost, LightGBM, and CatBoost.

Adaptive Boosting (AdaBoost) algorithm improves the accuracy of the classifier by weighting and repeated training of datasets. Yang et al. (2017) established the AdaBoost classification model. In comparison with logistic regression, decision tree, and neural network classification models, the results illustrate that the AdaBoost classification model demonstrates superior accuracy and robustness, with better practicability and wider application range. These methods use a new decision tree algorithm (AdaBoost), which greatly improves accuracy and robustness compared to traditional risk modelling algorithms.

GBDT (Gradient Boosting Decision Tree) learns a weak learner by fitting the residuals in each iteration, and the direction of the residuals is the globally optimal direction, that is, to train a weak learner in each round to minimize the sample loss. The principle diagram of GBDT is as follows Figure 2.1.

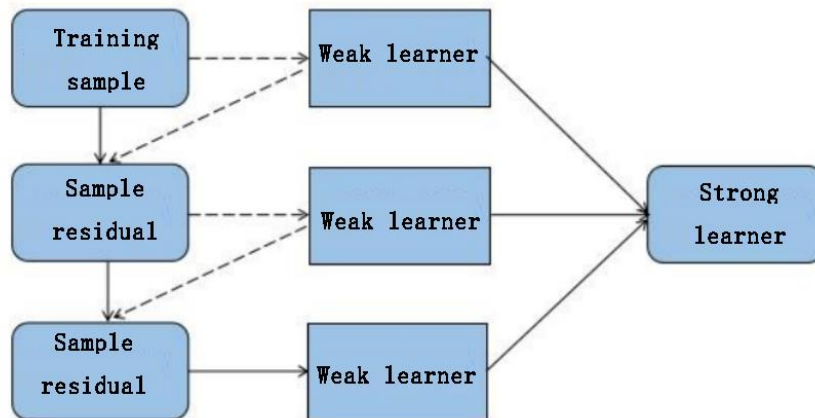


**Figure 2.1: Principle diagram of GBDT**

The primary advantage of Gradient Boosting Decision Trees (GBDT) is its high prediction accuracy, which is achieved without the need for special preprocessing or

normalization of the data. Due to the interdependency between weak learners, training data in parallel becomes challenging; GBDT (Gradient Boosting Decision Tree) cannot directly process text classification features, which need to be encoded in the pre-processing stage. Furthermore, Gradient Boosting Decision Trees (GBDT) trains the model by emphasizing the residuals of the samples, which may introduce gradient bias and consequently prediction bias. Considering the advantages and disadvantages of the GBDT calculation framework, several relatively effective algorithms have been developed, including XGBoost, LightGBM, and CatBoost algorithms (Demir & Sahin, 2023).

#### 2.2.3.4 XGBoost and LightGBM algorithm

XGBoost is an efficient and distributed gradient boosting algorithm that utilizes decision trees. Integrating multiple basis learners to form a strong learner, is also an optimization process of learning by using an additional model and forward step algorithm. Compared to the classical GBDT (Gradient Boosting Decision Tree), XGBoost has made some improvements by adding regular terms to the objective function and using second-order Taylor expansion to approximately replace its loss function. To enhance accuracy of model, XGBoost utilizes information from the second-order gradient of its own loss function. This improvement significantly enhances the effectiveness and performance of XGBoost. Additionally, by introducing Taylor second-order expansion and further refining the loss function, the XGBoost algorithm enhances prediction accuracy even further.

Chen and Guestrin (2016) give the ensemble learning algorithm XGBoost. Xia et al. (2017) employed the XGBoost model to assess the credit risk of commercial bank loans, showcasing its high accuracy and AUC (Area Under the Curve) performance in experimental results. Chongren et al. (2019) and team proposed an XGBoost model for credit risk assessment, utilizing Bayesian parameter optimization. This method uses the latest XGBoost

algorithm for risk modeling. Compared with the previous model, its accuracy rate has increased by more than 10%, which greatly improves accuracy of the risk control model. Moreover, its AUC (Receiver Operating Characteristic) value, F1-Score (Balanced Score) value, and other indicators are very good, but its operation and reading are very slow.

LightGBM (Light Gradient Boosting Machine) is a machine learning algorithm rooted in Gradient Boosting Decision Tree (GBDT) methodology, developed by Microsoft for its classification performance and speed. Its advantages: 1. Fast Training and Prediction Speed: LightGBM is known for its rapid training and prediction capabilities, making it suitable for large datasets and complex models. 2. Low Memory Consumption: The algorithm is designed to be memory-efficient, which helps in handling large-scale data without significant resource overhead. 3. Scalability: It scales well with the size of the data, enabling effective performance even with increasing data volumes. Disadvantages: 1. Sensitivity to Noise: LightGBM can be sensitive to noisy data, which may affect model performance and accuracy. 2. Need for Parameter Tuning: The algorithm requires careful tuning of hyperparameters to achieve optimal performance, which can be time-consuming. 3. High Data Dependence: The quality of the model heavily depends on the quality of the input data, making it crucial to have well-prepared and clean data. In short, LightGBM is a powerful tool for machine learning tasks, offering significant advantages in speed and efficiency. However, attention must be given to data quality and parameter tuning to mitigate its limitations.

LightGBM is an efficient and scalable machine learning algorithm designed to handle large-scale and high-dimensional datasets effectively (Ke et al., 2017). Therefore, many experts and scholars use it to study risk control models, Anjali et al. (2018) used a variety of ensemble learning of decision tree-based learners to study bank loan defaults, and the results

showed that the GBDT (Gradient Boosting Decision Tree) model had the best performance. The LightGBM model, optimized using Bayesian parameter optimization, demonstrates superior performance in credit risk assessment compared to traditional methods (Yazna et al., 2022). Ponsam et al. (2021) conducted experiments using the LightGBM algorithm in the risk control model, and the experimental results indicated that while the LightGBM algorithm boasts a very fast modeling speed, its accuracy falls slightly short of the XGBoost algorithm (Coşkun et al., 2021).

**Gradient Deviation (Demir et al., 2023):** Gradient deviation refers to the issue where the gradients used to update the model during training are not perfectly aligned with the true gradients of the loss function. This misalignment can occur due to various factors, including approximations made during optimization or the inherent nature of the gradient boosting algorithm. Both XGBoost and LightGBM, which are implementations of gradient boosting, can suffer from this issue, though they employ techniques to mitigate it.

**Prediction Deviation (Yao et al., 2022):** Prediction deviation involves discrepancies between the predicted values and the actual target values. This can arise from overfitting, model bias, or inefficiencies in handling complex relationships in the data. XGBoost and LightGBM, despite being powerful, can exhibit prediction deviation due to their reliance on decision trees and the inherent limitations of tree-based models in capturing intricate patterns.

**Gradient Deviation and Prediction Deviation (Ahn et al., 2023):** CatBoost, developed by Yandex, introduces several enhancements over XGBoost and LightGBM. It specifically addresses gradient deviation and prediction deviation issues through a technique called Ordered Boosting, which reduces the bias in gradient estimation and improves model accuracy. **Handling Categorical Features:** One of CatBoost's key innovations is its ability to handle categorical features directly without extensive preprocessing. CatBoost uses a

technique called "categorical feature encoding" that transforms categorical variables into numerical representations more effectively than traditional methods, preserving useful information and enhancing performance.

In summary, while XGBoost and LightGBM have their strengths, they can face challenges related to gradient and prediction deviations and do not handle raw text features directly. CatBoost addresses these issues by incorporating advanced techniques for both gradient boosting and categorical feature handling, making it a robust alternative in these contexts.

#### **2.2.4 Summary status of risk control model**

Credit risk assessment has traditionally relied on methods such as credit scoring models and statistical techniques like logistic regression. These methods provide a foundational understanding of a borrower's risk profile based on historical data and financial metrics. However, they have several limitations: 1. Assumptions and Simplifications: Traditional models often assume linear relationships and may not fully capture the complex, non-linear interactions among risk factors. For example, logistic regression assumes a linear relationship between predictor variables and the log-odds of default, which can be restrictive when dealing with real-world data exhibiting complex patterns. 2. Data Constraints: Traditional models are limited by the quality and scope of historical data. They may not adapt well to changes in economic conditions or borrower behavior, potentially impacting their accuracy during economic downturns or with new types of financial products. 3. Model Rigidity: The standard models may struggle with high-dimensional or non-linear data, leading to reduced performance in scenarios where new or unforeseen patterns emerge.

Recent advances in credit risk assessment have introduced machine learning (ML) and deep learning (DL) approaches, such as Random Forests, Support Vector Machines (SVMs),

Gradient Boosting Machines (GBMs), and Neural Networks. These techniques offer several advantages over traditional methods: 1. Handling Non-Linearity: ML and DL models can capture complex, non-linear relationships between variables more effectively than linear models. For instance, neural networks and SVMs can model intricate patterns in data that logistic regression might miss. 2. Improved Accuracy: Advanced techniques, such as ensemble methods and boosting algorithms (e.g., XGBoost and LightGBM), have demonstrated superior performance in terms of accuracy and robustness, often outperforming traditional models. 3. Adaptability: Modern algorithms are more adaptable to dynamic and evolving datasets. For example, GBMs and their variants can handle high-dimensional data and adapt to new patterns more effectively.

However, these advanced methods also face challenges: 1. Interpretability: Complex models like deep neural networks often lack transparency, making it difficult for stakeholders to understand and trust their predictions. This is a critical issue in regulatory environments where model explainability is essential. 2. Classification performance: Although the new machine learning algorithm has high accuracy, it requires a relatively long time for classification. Compared with logistic regression algorithms, it is not compatible with the advantages of classification performance. 3. Complexity: the use of new machine learning algorithms is very complicated, especially in the process of parameter adjustment.

To address these gaps, the proposed model incorporates the following features: 1. Find an algorithm with high accuracy and optimize it to improve interpretability. In the expert research mentioned above, XGBoost, LightGBM, and CatBoost algorithms have high accuracy, and the best algorithm can be selected among these algorithms for accuracy. 2. Process the data before entering algorithm testing to reduce the number of variables and improve the classification performance of the algorithm. 3. Simplify the complexity of

algorithm usage by combining multiple algorithms. This study can find an algorithm that automatically finds parameters to simplify the complexity of the model algorithm.

### 2.3 CatBoost Algorithm

CatBoost, developed by the Russian search giant Yandex in April 2017, builds upon the Gradient Boosting Decision Tree (GBDT) algorithm framework described in section 2.2.3.6. CatBoost is particularly effective at handling categorical features (Qi et al., 2021). It significantly enriches feature dimensions by combining categorical features and employs efficient strategies to reduce overfitting while utilizing the entire dataset for training, thus optimizing the use of data.

Set observation dataset, in this context, denotes the M-dimensional vector comprising both numerical and categorical features, with  $X$  representing the marker value. In the CatBoost algorithm, numerical features undergo binarization using the oblivious tree (Langley & Sage, 1994) as the base predictor. Floating-point features and statistics are binarized using one-hot encoding (Yu et al., 2022). Additionally, categorical features are transformed into numerical features through the following steps: 1. Random shuffling of observations to generate multiple random sequences. 2. Replacement of each class with the average target value of the training dataset for a given sequence.

$$X_{ik} = \frac{\sum_{j=1}^n [X_{jk} = X_{ik}] \cdot Y_j}{\sum_{j=1}^n [X_{jk} = X_{ik}]} \quad \text{Equation 2.1}$$

Where, if  $X_{ik} = X_{jk}$  then  $[X_{ik} = X_{jk}]$ , otherwise, if the same category value is encountered before the given value in the permutation, it is replaced with 0. 3. Set  $\theta = (\theta_1, \theta_2, \dots, \theta_n)$  the type characteristic values to be converted to numeric values:

$$X_{\phi,k} = \frac{\sum_{j=1}^{p-1} [X_{\theta_k} = X_{\phi,k}] Y_{\sigma_j} + a \cdot P}{\sum_{j=1}^{p-1} [X_{\theta_k} = X_{\phi,k}] + a} \quad \text{Equation 2.2}$$

The CatBoost algorithm introduces a prior value  $P$  and a parameter  $a$  ( $a > 0$ ) to mitigate noise in low-frequency categories. When dealing with feature combinations, CatBoost employs a greedy strategy: 1. The initial split of the tree does not involve any combinations. 2. In subsequent splits, all existing combinations and categorical features in the current tree are combined with all categorical features in the dataset, and the resulting combined value is immediately converted to a numerical value. 3. All splits chosen in the tree are treated as categories with two values, and these are used in combinations to generate a mixture of numerical values and categorical features.

To address gradient bias processing, the CatBoost algorithm constructs a tree in two stages: 1. Tree Structure Selection: It chooses the tree structure, determining the value of the leaf node after finalizing the structure. 2. Splitting Method Enumeration: Various splitting methods are enumerated, and the resulting tree is evaluated by computing the value of the leaf node to select the best segmentation. 3. Leaf Node Value Calculation: Values of the leaf nodes in this two-stage process are computed using gradient or Newtonian step approximations.

CatBoost synchronizes training datasets and efficiently handles categorical features, thereby improving feature processing efficiency. Additionally, its leaf node calculation algorithm effectively addresses overfitting and minimizes the need for extensive hyperparameter tuning, enhancing the model's versatility (Dina et al., 2022). Moreover, CatBoost binarizes floating-point features, statistics, and one-hot encoded features, enabling straightforward binarization of model output during the scoring process.

## 2.4 GWO algorithm

The Grey Wolf Optimizer (GWO), developed by Mirjalili et al. (2014), is an optimization algorithm based on population intelligence. It is celebrated for its robust convergence performance, simple structure, and ease of implementation, requiring minimal parameter adjustments. Moreover, it incorporates convergence factors and information feedback mechanisms, allowing for adaptive adjustments that enhance its effectiveness in solving optimization problems (Precup et al., 2016). Inspired by the hunting behavior of gray wolves, GWO offers an optimized search method that balances local optimization with global exploration. Its simplicity and minimal parameter tuning, combined with its adaptability, have led to successful applications in fields such as parameter optimization and image classification.

### 2.4.1 GWO algorithm principle and mathematical formula

The functioning of the Gray Wolf Optimization (GWO) is as follow: Gray Wolves belong to the canid family, exhibiting group living and occupying the apex predator position in the ecosystem. The wolves adhere to a rigid social dominance hierarchy, as illustrated in Figure 2.2.

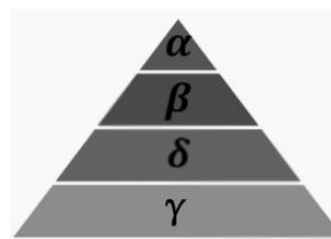


**Figure 2.2: GWO wolf pack level**

In the hierarchical structure of a wolf pack, the first level is occupied by the  $\alpha$  (alpha) wolf, often referred to as the  $\alpha$  wolf. As noted by Shi et al. (2021), the  $\alpha$  wolf holds a position of leadership and authority within the pack. It plays a central role in determining predation

strategies, habitat selection, resting times, and other collective behaviors. The  $\alpha$  wolf is regarded as ruling wolf, as other members of the pack defer to its decisions. Importantly, the  $\alpha$  wolf may not necessarily be the strongest physically, but it possesses exceptional management skills, making it the most capable leader within the pack.

The  $\beta$  (beta) wolf, also known as the  $\beta$  wolf, holds the second position in the social hierarchy within a wolf pack (Li & Liu, 2022). The  $\beta$  wolf serves as the assistant to the  $\alpha$  wolf and plays a supportive role in decision-making processes within the pack. If the alpha wolf becomes incapacitated or dies, the  $\beta$  wolf often emerges as the most suitable candidate to assume the role of the alpha wolf. While subordinate to the alpha wolf, the  $\beta$  wolf exerts significant influence over the rest of the pack's social hierarchy, effectively managing the interactions and dynamics within the group.

In the social structure of a wolf pack, the third-ranking position is held by the  $\delta$  wolf, playing a subordinate role to both the  $\alpha$  and  $\beta$  wolves. The  $\delta$  wolf holds authority over the remaining hierarchy within the pack. Typically,  $\delta$  wolves encompass a variety of individuals within the pack, such as young wolves, sentinel wolves, hunting wolves, old wolves, and suckling wolves.

Moving down hierarchy, the fourth level consists of  $\gamma$  wolves. These wolves are typically expected to adhere to the orders and commands of higher-ranking social classes. While the influence of  $\gamma$  wolves may appear limited, their presence is crucial for maintaining order within the wolf pack. Without  $\gamma$  wolves, internal problems such as cannibalism may arise.

The GWO algorithm operates through a sequence of steps inspired by the social hierarchy and hunting tactics of wolves:

#### 2.4.1.1 Social hierarchy model of wolves.

During the optimization process of the GWO algorithm, individuals in the population are assessed for fitness, with the top three individuals distinguished as  $\alpha$ ,  $\beta$ , and  $\delta$  wolves, while the rest are categorized as  $\gamma$ . This organization into four social classes  $\alpha$ ,  $\beta$ ,  $\delta$ , and  $\gamma$  mirrors the hierarchical structure observed in wolf packs. This division enables the algorithm to prioritize the most promising solutions represented by the  $\alpha$ ,  $\beta$ , and  $\delta$  wolves, guiding the optimization process in each iteration. These individuals play crucial roles in steering the algorithm towards better solutions, leveraging their superior fitness levels to influence the direction of the search for optimal solutions.

#### 2.4.1.2 Encircle the prey (Encircling Prey).

When a gray wolf searches for prey, it approaches and surrounds the prey, as shown in Figure 2.3. The mathematical model for this behavior is as follows:

$$D = C \cdot X_p^{(t)} - X^{(t)} \quad \text{Equation 2.3}$$

$$X(t + 1) = X^{(t)} - A \cdot D \quad \text{Equation 2.4}$$

Equation 2.3 is meaning: Distance between the individual and the prey. Equation 2.4 is meaning: Grey Wolf position update formula.

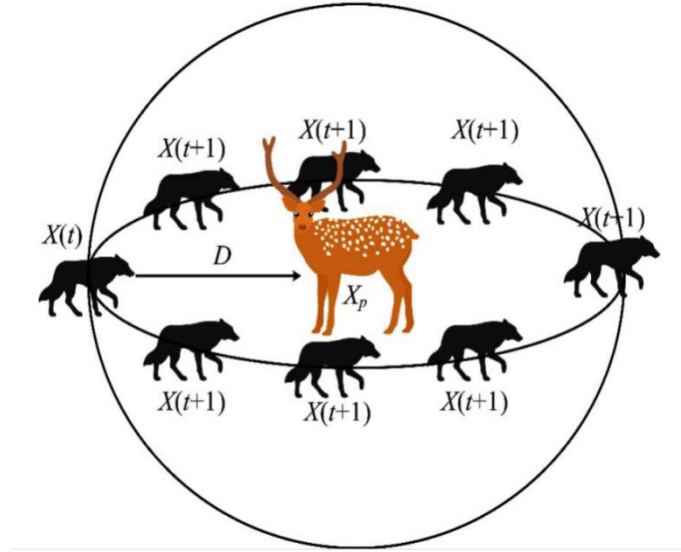


**Figure 2.3: Possible location of gray wolf surrounding prey**

Among them, "X": represents the position of the gray wolf, "t": the current position of the number of iterations, "X<sub>p</sub>" is the position of the prey,  $A = 2 \cdot \alpha \cdot r_1 - \alpha$ , and  $C = 2 \cdot r_2$  is the coefficient,  $r_1$  and  $r_2$  are  $[0,1]$ , random vectors  $\alpha$ :  $\alpha = 2 - t \cdot (2/T)$ , where  $T$  is the maximum number of iterations.  $A$  and  $C$  are synergy coefficient vectors, and  $\alpha$  decreases linearly from 2 to 0 during the entire iteration;  $r_1$  and  $r_2$  are random vectors in  $[0,1]$ .

To explain the meanings of the symbols in Formulas 2.3 and 2.4, see the detailed information in Table 2.1.

**Table 2.1: Parameters description**

| Symbolic name | Meaning                                                                 |
|---------------|-------------------------------------------------------------------------|
| X             | It represents the position of the gray wolf.                            |
| t             | It is the current position of the number of iterations.                 |
| T             | It is the maximum number of iterations.                                 |
| $\alpha$      | It is random vectors, Calculation formula: $\alpha = 2 - t \cdot (2/T)$ |
| $r_1, r_2$    | $r_1$ and $r_2$ are random vectors in $[0,1]$ .                         |
| A             | Calculation formula: $A = 2 \cdot \alpha \cdot r_1 - \alpha$            |

|       |                                      |
|-------|--------------------------------------|
| C     | Calculation formula: $C=2 \cdot r_2$ |
| $X_p$ | It is the position of the prey.      |

#### 2.4.1.3 Hunting

Gray wolves possess the remarkable ability to identify potential prey locations, which corresponds to optimal solutions in optimization problems. In their search process, they heavily rely on the guidance provided by the  $\alpha$ ,  $\beta$ , and  $\delta$  gray wolves. However, in many real-world problems, the spatial characteristics are often unknown, making it challenging for gray wolves to pinpoint the exact location of the prey, the optimal solution. To simulate the search behavior of gray wolves, it is assumed that the  $\alpha$ ,  $\beta$ , and  $\delta$  wolves exhibit a strong ability to identify potential prey locations. Hence, in each iteration of the optimization process, the positions of the best three gray wolves ( $\alpha$ ,  $\beta$ ,  $\delta$ ) are retained from the current population. The positions of the remaining search agents (including  $\gamma$  wolves) are then updated based on their position information. This approach allows for the exploration of candidate solutions across the search space. The mathematical model representing this behavior is as follows:

$$\begin{aligned}
X_1 &= X_\alpha - A_1 \cdot D_\alpha, & D_\alpha &= |C_1 \cdot X_\alpha - X|, \\
X_2 &= X_\beta - A_2 \cdot D_\beta, & D_\beta &= |C_2 \cdot X_\beta - X|, \\
X_3 &= X_\delta - A_3 \cdot D_\delta, & D_\delta &= |C_3 \cdot X_\delta - X|,
\end{aligned} \tag{Equation 2.5}$$

$$X_{(t+1)} = \frac{X_1 + X_2 + X_3}{3}$$

In the equation,  $X_\alpha$ ,  $X_\beta$ , and  $X_\delta$  denote the position vectors of the  $\alpha$ ,  $\beta$ , and  $\delta$  wolves in the current population, respectively.  $X$  represents the position vector of the gray wolf, while  $D_\alpha$ ,  $D_\beta$ , and  $D_\delta$  represent the distances between the current gray wolf candidate and the optimal three wolves. When the magnitude of  $|A|$  exceeds 1, the gray wolves disperse

widely across the area in search of prey. Conversely, when  $|A|$  is less than 1, the candidate solution positions tend to converge towards the random circular positions defined by  $\alpha$ ,  $\beta$ , and  $\delta$ , as illustrated in Figure 2.4. In summary,  $\alpha$ ,  $\beta$ , and  $\delta$  initially predict the approximate location of the prey (potential optimal solution), while the other candidate wolves update their positions randomly near the prey under the guidance of the current optimal wolf.

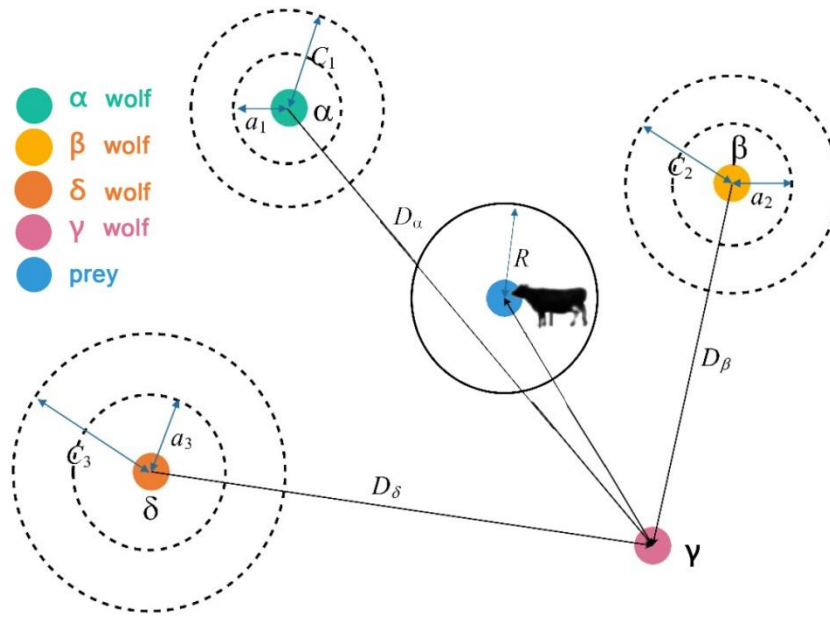


**Figure 2.4: Hunting method of wolves**

#### 2.4.1.4 Attacking prey (development stage)

As per Formula 6.5 described in the "Encircling Prey" stage, the reduction of the  $\alpha$  value directly influences the fluctuation of the  $A$  value. Essentially,  $A$  becomes a random vector within the interval  $[-\alpha, \alpha]$ , and its linear decrease occurs throughout the iterations. When  $A$  falls within the interval  $[-1, 1]$ , it implies that the subsequent position of the Search Agent can potentially lie anywhere between the current wolf and the prey.

Gray wolves execute their attack strategy when the prey ceases movement. To replicate the process of approaching the prey, the  $\alpha$  value diminishes gradually, thereby reducing the

range of fluctuations in  $A$ . Throughout the iterative phases, as the  $\alpha$  value linearly decreases from 2 to 0, the corresponding range of  $A$  values also adjusts within the interval  $[-\alpha, \alpha]$ .

As shown in Figure 2.5, when the value of  $A$  is within the interval, the gray wolf's next position can be anywhere between its current position and the hunting position (because it is random). When  $-1 < A < 1$ , the wolf pack attacks the prey (stuck in local optimum). When  $A > 1$  or  $A < -1$ , the gray wolf avoids the prey and searches other areas (to find the best solution).

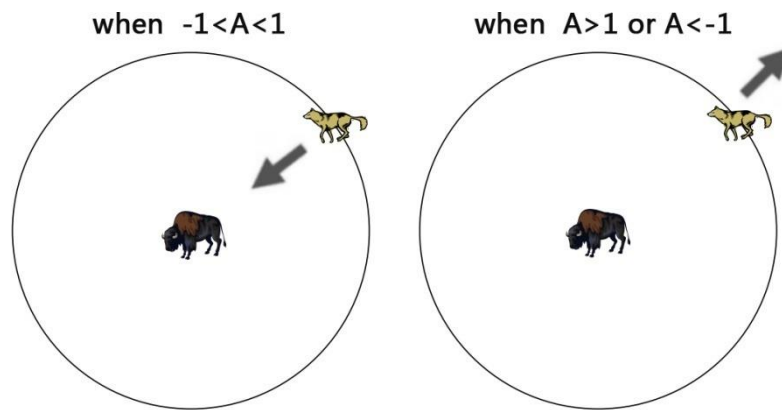


**Figure 2.5: Conditions for attacking prey**

#### 2.4.1.5 Search for prey (exploration phase)

Grey wolves coordinate their search for prey by referring to the positions of the alpha ( $\alpha$ ), beta ( $\beta$ ), and delta ( $\delta$ ) wolves. During their hunt, grey wolves initially spread out to explore independently before regrouping to converge on the prey. Leveraging the divergence model, a random value of  $|A|$  greater than 1 or less than -1 can be utilized to induce separation between grey wolves and their target. This emphasis on exploration facilitates the global optimization capabilities of the GWO algorithm.

Another coefficient, denoted as  $C$  ( $C=2 \cdot r_2$ ), is introduced, where  $r_2$  is a random vector within the range  $[0,2]$ . This coefficient represents the random weighting of a grey wolf's position with respect to the prey. A value of  $C$  greater than 1 indicates a stronger influence

weight, while a value less than 1 implies a weaker influence weight. The C value plays a vital role in preventing the algorithm from getting stuck in local optima and facilitating its ability to explore alternative solutions.

Unlike A, which decreases linearly over iterations, C introduces nonlinearity into the algorithm, thereby enhancing its global search capabilities throughout the optimization process. When the algorithm encounters difficulties in escaping local optima, the stochastic behavior of parameter C becomes crucial in guiding the search towards more promising regions of the decision space. This becomes particularly important as the algorithm progresses towards converging on the global optimal solution.

#### **2.4.2 GWO algorithm research status**

Based on the above introduction, the GWO algorithm possesses several advantageous characteristics. Firstly, it features a simple structure and requires minimal parameter adjustment, making it easy to implement. Furthermore, the algorithm incorporates convergence factors and information feedback mechanisms that are flexible and can be adjusted dynamically. This adaptability enables a harmonious balance between local optimization and global search, thereby enhancing its capability to thoroughly explore solution spaces.

The GWO algorithm stands out for its inherent traits, delivering impressive performance marked by superior solution accuracy and rapid convergence speed. Its ability to efficiently navigate complex optimization landscapes while maintaining robust convergence makes it a valuable algorithm in various optimization tasks.

The Grey Wolf Optimizer (GWO) algorithm boasts several advantages, such as its straightforward structure, easy implementation, and minimal parameter adjustments required.

Its novelty in algorithmic design and robust optimization capabilities, particularly when avoiding premature convergence, have captured the interest of many researchers. However, the practical complexity of optimization problems often poses challenges for the original GWO algorithm. To address this, numerous improved variants of GWO have been proposed to bolster its optimization prowess and overcome its limitations. Despite its promising performance, especially under optimal conditions, GWO's effectiveness across diverse problem landscapes remains a subject of ongoing research and refinement. Continued efforts to enhance the algorithm's adaptability and classification performance are crucial for its broader application in real-world optimization scenarios. In order to boost the algorithm's global convergence speed, dhavakumar and Gopalan (2021) introduced “chaos theory” into GWO, which is called CGWO. Karimi introduced an enhanced version of the GWO algorithm, known as EGWO, to effectively balance its exploration and exploitation capabilities (Karimi et al., 2018). Ibrahim introduced an enhanced variant of the GWO algorithm, known as COGWO2D, aimed at achieving a more balanced exploration-exploitation trade-off compared to the original algorithm (Ibrahim, 2017). To address the limitations of the GWO algorithm, Mohammad proposed modifications aimed at improving its slow convergence speed and low precision, GWO combined with the PSO algorithm (PSO-GWO) was proposed for the weighted distance optimization problem (Mohammad et al., 2021). The grey wolf weighted distance optimization algorithm (WD-GWO) was proposed, with enhancement entails employing the weighted sum of the optimal positions rather than a basic average to adjust a single position (Qu et al., 2018). Simulation experiments verify the superiority of the algorithm. A novel extension of the GWO algorithm, termed PGWO, was introduced, integrating Powell's local optimization method (Hu et al., 2018). Powell's method for local optimization is a technique used to compute the minimum

value of a function when its differentiability is unavailable. The algorithm consistently demonstrates its effectiveness through experimental validation. Powell's method finds practical application across a wide array of fields, including but not limited to machine learning, engineering, wireless sensor networks, environmental modeling, medicine, bioinformatics, and image processing. For example, Kauffmann proposed sequential hybridization of GWO and PSO to optimize single-region unit combination problems (Kauffmann et al., 2023). However, the method converges slowly due to long sequence execution time. (Mahdad & Srairi, 2015) have combined GWO with a local search algorithm to address secured smart grid power system management in emergencies. Compared with other optimization methods, the results show a promising application of this method.

## **2.5 Research and discussion of new algorithm**

According to the study by Tikhamarine et al. (2018). GWO has the following advantages: 1. Efficiency: GWO is computationally efficient and requires only a few parameters to be tuned. This makes it suitable for optimizing parameters of other algorithms where the parameter space can be large and complex. 2. Balance of Exploration and Exploitation: GWO effectively balances exploration (searching new areas) and exploitation (refining existing solutions). This balance helps in finding the global optimum while avoiding local optima. 3. Adaptability: GWO is adaptable to various problem types and dimensions. Its general nature allows it to be applied to different optimization problems without significant modifications. 4. Ease of Use: GWO has a straightforward implementation with minimal parameters, which simplifies its use in practice. The algorithm's core operations are relatively simple compared to some other optimization techniques. According to the content of 2.4, parameters of the GWO Algorithm: 1. Number of Wolves (Population Size): The number of candidate solutions (wolves) used in the

optimization process. This parameter affects the diversity and convergence speed of the algorithm. 2. Maximum Number of Iterations: The number of iterations the algorithm will perform. This parameter determines how long the algorithm will run and affects convergence. 3. Coefficient Vectors:  $\alpha$ : A coefficient that decreases linearly from 2 to 0. It controls the exploration and exploitation balance.  $r_1$  and  $r_2$ : Random vectors used to create diversity in the search process. Therefore, the GWO algorithm exhibits a simple structure, ease of use, and a small number of parameters to adjust. It is suitable for finding parameters for other complex algorithms. On the other hand, although the XGBoost algorithm achieves high accuracy, its classification speed is very slow. One requirement for optimizing risk control models is that the algorithm must have a fast classification speed. Thus, XGBoost cannot be used for optimizing risk control models. The CatBoost algorithm is known for its high precision, some degree of interpretability, and stability; however, it requires finding optimal parameters to achieve high accuracy. Therefore, this study selects the GWO and CatBoost algorithms to enhance the classification performance(speed) of the model.

In the discussion of the GWO optimization algorithm, researchers found that there is still some optimization space regarding the number of leaders missing from GWO. Therefore, this study proposes an adaptive GWO (EB-GWO), which balances the exploration and development capabilities of GWO (Thamaraiselvi et al., 2024). This approach aims to enhance the original Grey Wolf Optimizer (GWO) algorithm by leveraging its strengths to optimize parameters in the CatBoost model. By addressing the limitations of GWO's expansive exploration capacity, the method refines GWO through the incorporation of adaptive leadership (EB). This combined approach aims to enhance model convergence, reduce parameter complexity, and streamline model operations. By doing so, the approach mitigates the inefficiencies inherent in GWO and strikes a balance between local and global

search capabilities, thereby enhancing the model's accuracy. The refined algorithm's effective balance between local and global search enhances the accuracy of the CatBoost loan prediction model significantly. However, the CatBoost algorithm's feature processing may pose challenges, especially with a large number of features, resulting in substantial memory consumption. Thus, prior to employing the EBGWO algorithm for parameter optimization, it is crucial to conduct dimensionality reduction operations. The IV (Information Value) calculation can help reduce the feature set, ensuring smoother processing and optimal performance of the CatBoost algorithm.

This study introduces an adaptive version of the Grey Wolf Optimizer (EB-GWO) to improve its exploration and exploitation balance. This enhanced GWO algorithm uses adaptive leadership to address GWO's limitations and optimize parameters for the CatBoost model, leveraging GWO's strengths more classification performance.

## **2.6 Risk control model data processing method**

To enhance the accuracy and discriminative capability of the model, this section needs to refer to the data processing method of risk control. Of course, datasets may not be processed in a risk-controlled manner, which may affect the final accuracy. The main contents of the data processing method include data processing, WOE (Weight of Evidence) partitioning and IV value calculation, etc.

### **2.6.1 Data processing**

Data processing mainly consists of the following aspects: 1. Missing values; 2. Outlier processing; 3. Good and bad customer definition; 4. Sample size; 5. Define the sample window period;

2.6.1.1 Missing values: Missing values can impact calculations and lead to errors in experiment programs. It's crucial to address them properly. Two methods, single imputation and multiple imputations, offer approaches to manage missing values efficiently in one step.

Single imputation is used to construct an imputable value for each missing value. Commonly used single-value imputation methods include mean imputation, regression imputation, median imputation, mode imputation, etc. The mean imputation method refers to the imputation of the mean value of the missing attribute. The regression imputation rule is a method of missing data imputation by establishing a regression model with non-missing values as independent variables and missing values as independent variables.

Multiple imputations were proposed by Rubin in 1987 and became one of the most popular methods to deal with missing data in recent years. Instead of filling in the missing values with a single value, multiple imputations replace the missing values with  $m$  ( $m \geq 2$ ) plausible imputations, resulting in  $m$  complete datasets (the  $m$  imputations are randomly drawn from the posterior probability distribution of the missing values according to Bayesian theory). When the missing rate is relatively low, a small value of  $m$  can achieve a good imputation effect, while when the missing rate is high, a larger value of  $m$  is needed to improve the imputation effect. After the imputation is completed, standard data analysis methods can be used to analyze each complete dataset, and finally, the results are synthesized.

2.6.1.2. Outlier processing: An outlier, also known as an outlier, is a value that is particularly large or small compared to most values in a data series, and need processing.

2.6.1.3. Good and bad customer definition: There are many kinds of good and bad customer definitions of the risk control model, this thesis adopts the most classic definition according to the length of the overdue definition (Noh et al., 2005). The criteria are as

follows: 1. M3+: M stands for month and M3+ stands for customers who are more than 90 days overdue (gold standard scheme). 2. M2+: M represents the month, and M3+ represents the customer who is overdue for more than 60 days (compromise solution). 3. M1+: M represents the month, and M3+ represents the customer who is overdue for more than 30 days (common scheme).

If the performance period is longer than 6 months, Maxpd30 can replace Maxpd60 or Maxpd90, which means that if the number of bad customers is too small, the time can be extended. (Maxpd30 means 30 days overdue; Maxpd60 is 60 days overdue. Maxpd90 means 90 days).

Traditional financial companies commonly define bad customers when the criteria measured is above M1+. So, assessing through the traditional model to identify good and bad customers, we can identify a lot of good customers, but also ignore a lot of potential good customers. Therefore, the model needs to be optimized and improved in this aspect. Therefore, this thesis changes the definition of bad customers to the standard of M3+(more than 90 overdue) to objectively improve the recognition rate of good customers and bad customers.

2.6.1.4. Sample size: There is no fixed standard, it depends on data quality, algorithm selection, etc.

2.6.1.5. Window period and performance period: The "observation period" is the time window for collecting variables and features, and the "performance period" is the time window for collecting whether to trigger the definition of bad samples.

### 2.6.2 WOE binning method

According to experts (Sharma et al., 2023). WOE (Weight of Evidence) serves as a method for encoding original independent variables, while data boxing involves partitioning continuous data into multiple groups or categorizing discrete data into features with reduced categories. Then, the WOE value of each bin is calculated to obtain a more accurate data distribution. It is a statistical technique often used in credit scoring and risk modeling. It helps quantify the relationship between predictor variables and the binary outcome variable by assigning weights to different levels or categories of predictors. Here's a quick rundown of the concept:

WOE transforms categorical variables into a format that can be used effectively in logistic regression models. It helps in understanding the impact of each category on the target variable. WOE is calculated as follows:

$$WOE_i = \ln\left(\frac{p_{y1}}{p_{y0}}\right) = \ln\left(\frac{B_i}{B} / \frac{G_i}{G}\right) = \ln\left(\frac{B_i}{G_i} / \frac{B}{G}\right) \quad \text{Equation 2.6}$$

Where  $p_{y1}$  is the proportion of bad samples in all bad samples,  $p_{y0}$  is the proportion of good samples in all good samples.  $B$  is the total number of bad samples,  $B_i$  is the number of bad samples corresponding to the variable,  $G$  is the total number of good samples corresponding to the variable,  $G_i$  is the number of good samples corresponding to the variable. Note: A model target variable  $y$  of 1 is denoted as a default user (bad sample), and a target variable of 0 is denoted as a normal user (good sample).

The benefits of WOE: 1. Monotonicity: WOE helps in creating monotonic relationships between the predictor and the target variable, which is useful for logistic regression models. 2. Handling of Categorical Variables: WOE is effective in dealing with categorical variables

by converting them into continuous variables. 3. Outlier Management: It can reduce the impact of outliers by aggregating them into broader categories.

According to experts (Stratman & Brandenburger, 2018), traditional risk control models use equal-width binning. This method is simple and straightforward, but it has the disadvantage of data often concentrating within certain intervals, leading to nearly identical bin numbers and significant information loss. Additionally, it usually requires some level of expert knowledge. According to research by Matteucci et al. (2024), equal-frequency binning can evenly distribute the data and avoid information loss. The advantage of equal-frequency binning is that when data is concentrated within a specific interval, the bin numbers do not become identical, thus overcoming the drawbacks of equal-width binning. (See WOE binding file in Appendix B for Python code).

Chi-square tests are used to evaluate the association between categorical features and the target variable in a dataset. Specifically: 1. Chi-Square Test: This statistical test assesses whether there is a significant association between two categorical variables. It compares the observed frequencies in each category of a contingency table with the expected frequencies if there were no association between the variables. 2. Chi-Square Boxes: In the context of feature selection or preprocessing, "Chi-square boxes" may refer to the Chi-square statistic values calculated for each feature variable against the target variable. These values indicate how strongly each feature is associated with the target. 3. Usage: By calculating Chi-square statistics for each categorical feature, you can identify which features have a significant association with the target variable. This helps in feature selection and understanding the relationship between variables. In practice, the Chi-square test is used to assess the independence of categorical variables and to select features that have a strong relationship with the target.

### 2.6.3 Screening variable

IV (Information Value), utilized to assess the predictive capability of a variable, indicates the amount of information it contributes to the prediction. Generally, higher IV values suggest stronger predictive abilities, making the variable more likely to be selected for modeling.

The computation of IV relies on the Weight of Evidence (WOE) value. The variable is divided into bins, and then the IV value of each group of box data is summed up, and finally, the iv value of a variable is obtained. The calculation formula is as follows.  $Bad_i$  refers to the number of bad customers and  $Good_i$  refers to the number of hospitable customers.  $Bad_T$  is the total number of bad customers and  $Good_T$  is the total number of good customers.

$$\begin{aligned} IV_i &= \left( \frac{Bad_i}{Bad_T} - \frac{Good_i}{Good_T} \right) * WOE_i \\ &= \left( \frac{Bad_i}{Bad_T} - \frac{Good_i}{Good_T} \right) * \ln \left( \frac{Bad_i / Bad_T}{Good_i / Good_T} \right) \end{aligned} \quad \text{Equation 2.7}$$
$$IV = \sum_{i=1}^n IV_i$$

From the formula 2.7, it can be seen that a larger IV value indicates a stronger correlation. Traditional risk control models that use logistic regression algorithms do not typically incorporate the calculation of the IV value for variable selection. This study proposes an optimization approach for variable screening: first calculate the IV value and then filter the variables based on this value. The screening criteria are as follows:

According to research by Mushava and Murray (2022), if  $IV < 0.02$ , the variable has very little association and can be removed. If  $0.02 < IV < 0.1$ , the variable shows some level of association and can be considered for inclusion in the model. If  $0.1 < IV < 0.3$ , the variable has an impact on the model results, though it is relatively weak, so it should be retained. If

$0.3 < IV < 0.5$ , the variable significantly affects the model results and should be retained and closely monitored. If  $IV > 0.5$ , the variable has a very strong association, which may indicate overfitting; therefore, it is recommended not to include this variable in the model.

Mutual Information (Shannon, 1948), abbreviated to MI. Mutual information measures the amount of information obtained about one variable through another. It quantifies the reduction in uncertainty of one variable given the knowledge of another, indicating the degree of dependency between the two variables. Higher mutual information suggests a stronger relationship between the variables.

## **2.7 Chapter Summary**

This chapter mainly describes the algorithmic literature about risk control models and the processing methods of risk control datasets.

## **CHAPTER 3**

### **RESEARCH METHODOLOGY**

#### **3.1 Overview**

This chapter outlines the research methodology employed to address the identified research problem and questions. The research methodology is designed to ensure that the approach is systematic and targeted towards solving the core issues highlighted in the study. The chapter is organized as follows: 3.1.1 Research Problem and Questions. 3.1.2 Methodological Process.

##### **3.1.1 Research Problem and Questions**

This subsection provides a clear problem statement and research questions derived from the literature review, emphasizing the significance of the chosen methodology.

The research problem addressed in this study arises from the need to improve predictive accuracy and explainability in credit risk assessment models. Existing methods, such as logistic regression, have limitations in terms of predictive performance and interpretability. This study aims to develop and optimize advanced machine learning models to address these shortcomings. Problem Statement: Traditional credit risk models often suffer from issues related to predictive accuracy and explainability. Specifically, while models like logistic regression provide a baseline for risk assessment, they may not fully capture complex relationships in credit data and can lack interpretability. This research seeks to develop a more accurate and interpretable model by leveraging advanced machine learning algorithms and optimization techniques. Research Questions: 1. RQ1: How can advanced machine learning algorithms, such as XGBoost, LightGBM, and CatBoost, be utilized to improve

predictive accuracy in credit risk assessment compared to traditional models like logistic regression. 2. RQ2: What are the limitations of the CatBoost algorithm in credit risk modeling, and how can these limitations be addressed through algorithm optimization and parameter tuning. 3. RQ3: How can the explainability of the CatBoost model be enhanced to provide clearer insights into the decision-making process, and what role does the SHAP framework play in this enhancement.

### 3.1.2 Methodological Process

This subsection describes the methodology in detail, including data situation, algorithm selection and optimization, and evaluation methods, as outlined in Section 3.2.

The research methodology is designed to address these questions through a structured approach: **Data Situation:** Detailed descriptions of datasets from China\_Credit and LendingClub are provided to ensure that the models are tested on comprehensive and representative data. **Algorithm Selection and Optimization:** Three advanced algorithms are evaluated and optimized. The selection process involves comparing these algorithms based on their performance metrics, and optimization includes addressing specific limitations identified in the literature. **Algorithm Evaluation Method:** A rigorous evaluation framework is employed, including confusion matrix, accuracy metrics, ROC curve, and AUC, to assess the performance of the models and ensure they meet the desired criteria for predictive accuracy and explainability. By linking the methodology directly to the research problem and questions, this chapter ensures that the approach is accountable and specifically designed to address the identified issues in credit risk assessment.

### 3.2 Research Methodology

The process of this research method is outlined in Figure 3.1. Phase1 Problem Definition and Dataset, this Phase consists of two parts:1. Problem Definition; 2. Dataset.1. Problem Definition, is where the relevant previous works are reviewed in order to apprehend the current issues of the problem domain and the challenges in the state-of-the-art approaches are identified based on the extensive review of approaches reported by other researchers. Subsequently, the challenges and issues are identified and are later on translated into problem statements and research objectives of the research work. This section of material corresponds to chapters 1 and 2. 2. Dataset, this part introduces the details of the dataset, including how the data being collected., in order to benchmark the performance of the proposed algorithm. This corresponds to Section 3.2.1 in Chapter 3.

Phase 2 Algorithm Choices and Optimization: This part introduces the selection and Optimization of the algorithm. First, three algorithms (XGBoost, LightGMB, and CatBoost) with relatively high algorithm accuracy are selected for discussion, analysis, and experimental comparison and verification. Then, CatBoost algorithm was chosen to replace the old risk control model algorithm (logistic regression algorithm) to improve the accuracy of the model, which will be introduced in detail in Chapter 4. Secondly, CatBoost algorithm itself has third major shortcomings, the first shortcoming is that the algorithm classification performance needs to be further improved, which will be introduced in detail in Chapter 5. second disadvantage is that CatBoost requires manual parameter adjustment, which increases the complexity of the algorithm; At the same time, the good and bad parameters of the algorithm directly affect its accuracy. In this study, GWO algorithm is selected to automatically optimize this shortcoming of CatBoost algorithm. However, GWO algorithm itself also has defects and needs further optimization, which will be introduced in

detail in Chapter 5. Then, the combination of optimized GWO algorithm (EBGWO) and CatBoost algorithm is introduced to form a new algorithm (EBGWO-CatBoost), which solves the second shortcoming of CatBoost algorithm, that is, the complexity of the algorithm is simplified, at the same time, the accuracy of the algorithm can be further improved, which will be introduced in chapter 6. Finally, the third disadvantage is the insufficient explainability of the algorithm. This part will further improve the explainability of CatBoost algorithm with the help of the SHAP framework, which will be introduced in Chapter 7.

Phase 3 Validation: This part introduces the evaluation of algorithms, evaluates the algorithms proposed in the thesis, and verifies their comprehensive performance. This part is included in Chapter 4, Chapter 5, Chapter 6, and Chapter 7.

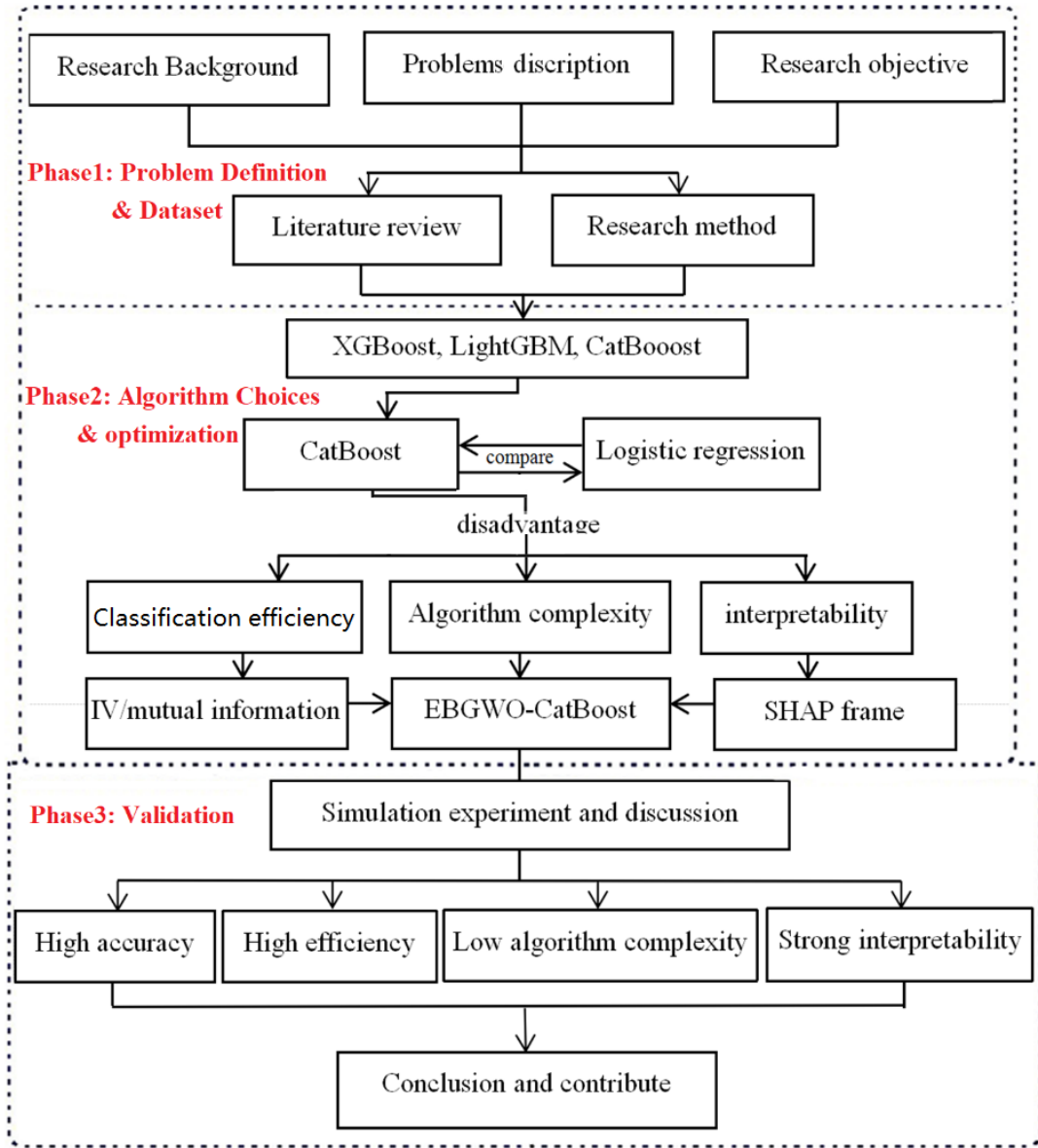


**Figure 3.1: Research Process**

### 3.2.1 Problem Definition and Dataset

At Phase 1, Problem Definition is where the relevant previous works are reviewed in order to apprehend the current issues of the problem domain and the challenges in the state-of-the-art approaches are identified based on the extensive review of approaches reported by other researchers. Subsequently, the challenges and issues are identified and are later on translated into problem statements and research objectives of the research work. This section

of material corresponds to Section 1 and 2. The following describe the dataset that will be used in the study.

#### 3.2.1.1 China\_Credit Dataset

The Chinese government started to collect personal credit information late. In January 2006, it centralized a unified personal credit information foundation database and just began to network formally to run. In this database, credit information collected by financial institutions with personal credit information will be entered into the personal credit file public platform established by the People's Bank of China in a unified manner, which realizes network interconnection and electronic management to facilitate mutual inquiry and information sharing. By the end of 2022, the basic database of the People's Bank of China had established credit files for nearly 970 million instances. The personal credit database is based on the credit information of the bank, including three parts: personal records, business records, and social public information records. China's credit system is not perfect, and it is impossible to effectively query the real credit status of customers. Most of the traditional financial institutions adopt the risk control model of logistic regression algorithm. Between good and bad customers, this model is directly denied loans. China is in a special situation, with about 16% of the world's population. More people than any other country. Chinese culture is quite different from that of other countries. As a result, the data on risky loans generated in China is quite different from those generated in other countries (Orgad et al., 2024). Therefore, experiments need to be conducted separately for the Chinese dataset.

The personal credit dataset released by the Chinese government can be divided into five parts according to the characteristics of personal credit, namely Character, Capacity, Capital, Collateral, and Condition ( Nawai & Shariff, 2010).

Chinese financial firms have largely taken the same tack, collecting data through three types of card information (Li et al., 2018). The first is the A-card (application Scorecard), which is the quantitative evaluation of the loan applicant in the approval phase, as shown in Figure 3.2.

| Attribute (variable)                      | Value / Range       | Points |                                     |
|-------------------------------------------|---------------------|--------|-------------------------------------|
| Base Points                               |                     | 220    | <input checked="" type="checkbox"/> |
| Annual Income                             | < \$25,000          | 55     | <input type="checkbox"/>            |
|                                           | \$25,000 - \$50,000 | 75     | <input type="checkbox"/>            |
|                                           | >\$50,000           | 90     | <input checked="" type="checkbox"/> |
| Years at Address                          | < 2 Years           | 45     | <input type="checkbox"/>            |
|                                           | > 2 years           | 63     | <input checked="" type="checkbox"/> |
| Employment Status                         | Full Time           | 63     | <input checked="" type="checkbox"/> |
|                                           | Other               | 49     | <input type="checkbox"/>            |
| TOTAL (if greater than 400; then approve) |                     | 439    |                                     |

| Attribute (variable)                      | Value / Range       | Points |                                     |
|-------------------------------------------|---------------------|--------|-------------------------------------|
| Base Points                               |                     | 220    | <input checked="" type="checkbox"/> |
| Annual Income                             | < \$25,000          | 55     | <input type="checkbox"/>            |
|                                           | \$25,000 - \$50,000 | 75     | <input type="checkbox"/>            |
|                                           | >\$50,000           | 90     | <input checked="" type="checkbox"/> |
| Years at Address                          | < 2 Years           | 45     | <input type="checkbox"/>            |
|                                           | > 2 years           | 63     | <input checked="" type="checkbox"/> |
| Employment Status                         | Full Time           | 63     | <input checked="" type="checkbox"/> |
|                                           | Other               | 49     | <input type="checkbox"/>            |
| TOTAL (if greater than 400; then approve) |                     | 439    |                                     |

**Figure 3.2: Data collection card**

The second is the B-card (behavior scorecard) post-loan management, this process predicts the borrower's future repayment ability and willingness by analyzing their repayment and transaction behavior in conjunction with other dimensions of data. The third is the C card (collection scorecard), which is collection management. When the current repayment state of the stage person is overdue, the probability of the loan becoming bad debt in the future is predicted, and the subdivided models such as rolling rate, repayment rate, and lost contact rate are derived. Data is collected through these three kinds of cards.

Personal credit has the characteristics of personalization, sociality, externality, accumulability, measurability, and constraint (Shang & Chiu, 2020). This process encompasses various dimensions of data to reflect individuals' social relationships, personal information, family dynamics, and professional qualifications. It also considers aspects of

human nature influenced by external factors such as moral conduct, reward, and punishment records. Moreover, the analysis entails scrutinizing transactional data and historical performance records with other economic entities, including financial institutions, public utilities, and tax agencies. Additionally, it evaluates indicators of willingness to pay, such as living conditions and income status (Jacques, 2023). Therefore, to better prevent financial risks and improve the profits of financial institutions, some typical datasets are specially processed and shared with financial companies, so that more financial institutions can use these datasets to conduct modeling research.

In 2020, a total of 107,115 records of data<sup>1</sup>, including account balance, credit term (month), previous credit payment status, target credit amount, and a total of 20 features have been collected (Wu et al., 2019).

#### 3.2.1.2 LendingClub dataset

LendingClub is the world's largest peer-to-peer Internet lending platform. Its main business is to assess the default risk of borrowers and set different borrowing rates based on the borrower's past credit history and other information. Borrowers can quickly obtain loans by submitting applications. Investors gain interest or income by reviewing the borrower's past credit history and borrowing purpose and deciding whether to lend money to borrowers with different borrowing rates. The loan dataset collected online on 2.92 million Americans (one in 10 of the population) over 13 years, includes age, income, job title, geographic location, loan purpose, credit rating, and more. The LendingClub dataset can be used for our

---

<sup>1</sup><http://archive.ics.uci.edu/ml/machine-learning-databases/statlog/german/>

economic, demographic, social, job, and political data analysis and user profiling of Americans.

LendingClub 2020 has a total of more than 128,000 data and 110 variables<sup>2</sup>. With a large amount of data and rich variables, the Lending Club is an ideal dataset for machine learning modeling on various algorithm experiments (Misheva et al., 2021). Financial research institutions can be used for data cleaning, variable screening, parameter adjustment, multi-algorithm comparison, unbalanced data processing, and other tests.

### **3.2.2 Algorithm Choices and Optimization**

In Phase 2 Algorithm Choices and Optimization. Experiment procedure: 1. The two datasets in Chapter 3, Section 3.2.1, were selected for test validation. The data processing method is referred to as the method in Section 2.6 of Chapter 2. According to the research by experts (Paul & Basu, 2024), the main methods for handling outliers include statistical methods ( $3\sigma$  rule, z-score) and PCA (Dimensionality reduction) based reconstruction error methods, which can effectively detect outliers. This study references these experts' research, First, the missing values and abnormal values are processed. To define good or bad customers and sample window period; Secondly, the calculation of IV (Information value) is carried out. According to research by Matteucci et al. (2024), using the equal frequency binning method allows for uniform data partitioning and helps avoid information loss. Finally, the sample size is selected. 2. Divide processed data, use 70% training data to train the model, and the remaining 30% data to test the trained model. 3. Selection of algorithms. In this study, XGBoost, Catboost, and LightGBM algorithms were selected for model

---

<sup>2</sup><https://www.lendingclub.com/investing/peer-to-peer>

training and testing. The algorithm with the best comprehensive performance was selected from these three algorithms, and compared with the algorithm in the original model (logistic regression), and the Catboost algorithm with the highest comprehensive performance was finally selected. This is covered in Chapter 4. 4. Optimize the filtered algorithm (Catboost algorithm).

In risk analysis, CatBoost, LightGBM, and XGBoost achieve high accuracy due to several expert opinions (Ahn et al., 2023) key reasons: 1. Advanced Boosting Techniques: Gradient Boosting: All three models use gradient boosting, which builds a series of decision trees where each tree corrects the errors of the previous ones. This iterative approach helps in refining predictions and improving accuracy. Error Correction: By focusing on the residuals or errors of previous models, these algorithms effectively address the complex patterns in risk-related data, leading to high accuracy in predicting loan defaults and other risk factors. 2. Regularization: XGBoost: Incorporates both L1 (Lasso) and L2 (Ridge) regularization to control the complexity of the model and avoid overfitting. This is particularly useful in risk modeling, where overfitting to historical data can lead to poor generalization on new data. LightGBM: Provides regularization parameters (`'lambda_l1'`, `'lambda_l2'`) to control the complexity of the model and prevent overfitting. Its efficient implementation allows for the handling of large datasets, which is common in risk assessment scenarios. 3. Efficient Data Handling: LightGBM: Utilizes a histogram-based approach that discretizes continuous features into bins, which speeds up training and reduces memory usage. This is especially beneficial when dealing with large datasets typical in risk control. XGBoost: Employs techniques like sparse matrix handling and parallel processing to efficiently handle large datasets and complex feature spaces, which helps in scaling risk models to large volumes of data. 4. Categorical Feature Handling: CatBoost: Designed to

handle categorical features directly by encoding them into a numerical format that preserves their ordinal relationships. This is particularly useful in loan risk models where categorical variables (e.g., job titles, regions) are prevalent and significant for risk assessment. Feature Interactions: CatBoost also effectively captures interactions between categorical features and other variables, which can enhance the model's ability to predict risk. 5. Robust Training Algorithms: CatBoost: Uses ordered boosting and other techniques to reduce overfitting and improve generalization. This is crucial for risk modeling where accurate prediction is vital to avoid false positives or negatives. LightGBM: Employs leaf-wise growth strategy, which allows it to capture more complex patterns compared to level-wise growth, making it effective in capturing intricate relationships in risk data. XGBoost: Features like tree pruning and early stopping contribute to its robustness. Tree pruning helps in reducing model complexity, while early stopping prevents overfitting by halting training when performance on a validation set deteriorates.

The high accuracy of CatBoost, LightGBM, and XGBoost in risk loan control models can be attributed to their advanced boosting techniques, efficient data handling, robust training algorithms, effective regularization, and ability to handle complex and imbalanced data. These factors enable the models to build strong predictive frameworks that can effectively assess and manage risk in loan control scenarios.

The algorithm choices and optimization process of the algorithm is shown in Figure 3.3. The process consists of two parts.

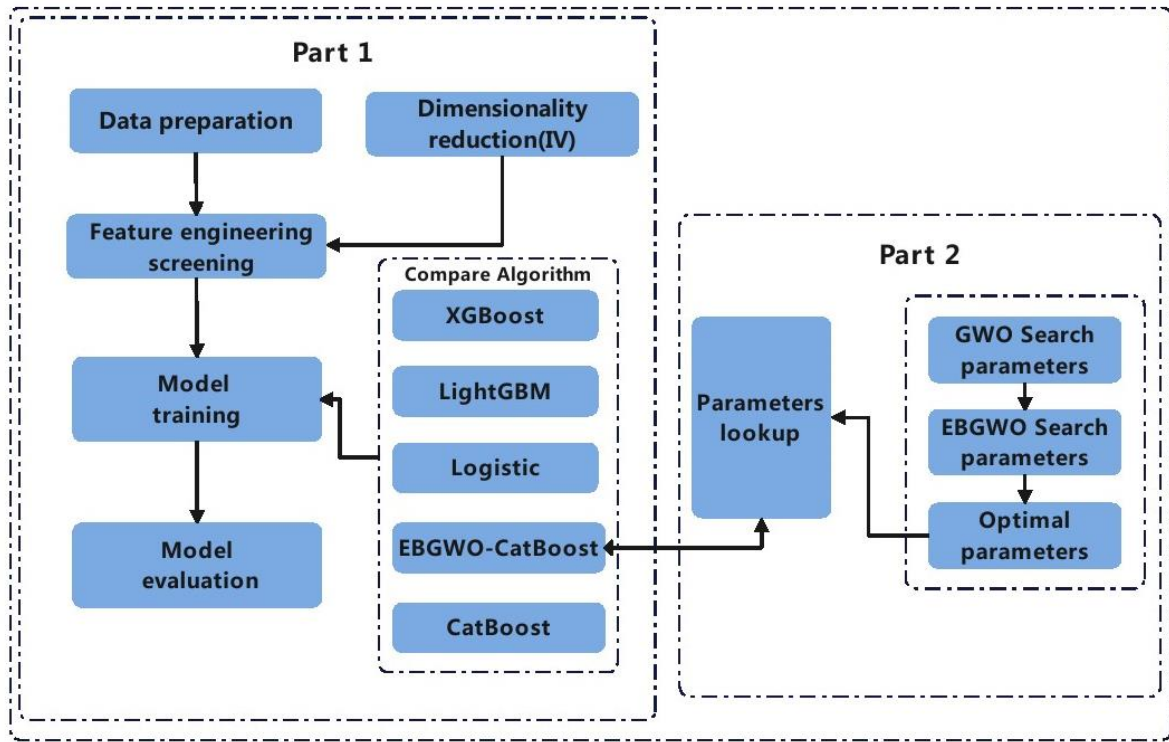


**Figure 3.3: Combined Algorithm Structure**

Part 1 is the selection process of the algorithm. Firstly, the data processing process is shown in Figure. 3.4. The specific process is as follows: 1. For data outliers, this study selects "residual map" to identify outliers, which can detect outliers of multiple attribute values and effectively identify outliers (Tang et al., 2022), then identified outliers data is deleted. 2. For missing values, this study adopts the multiple imputation method, which is reliable in processing missing values because each complete dataset is analyzed and then imputed (Fogarty et al., 2006). 3. For the definition of bad customers, this study adopts the definition method of M1+ (overdue for more than 30 days), because this time (M1+) is more reliable in the application of risk control model (Wang et al., 2021). 4. 1-year time observation period was adopted in this study, because according to (Ignatieva et al., 2016), the observation time of risk control model is the most suitable one year (VenkateswaraRao et al., 2023). 5. Data WOE Binning. 6. Finally, feature engineering is carried out, and the features with low

correlation degree are screened by IV (Information Value) value, and then the features with low correlation degree are deleted. Finally, Mutual information is used for verification.

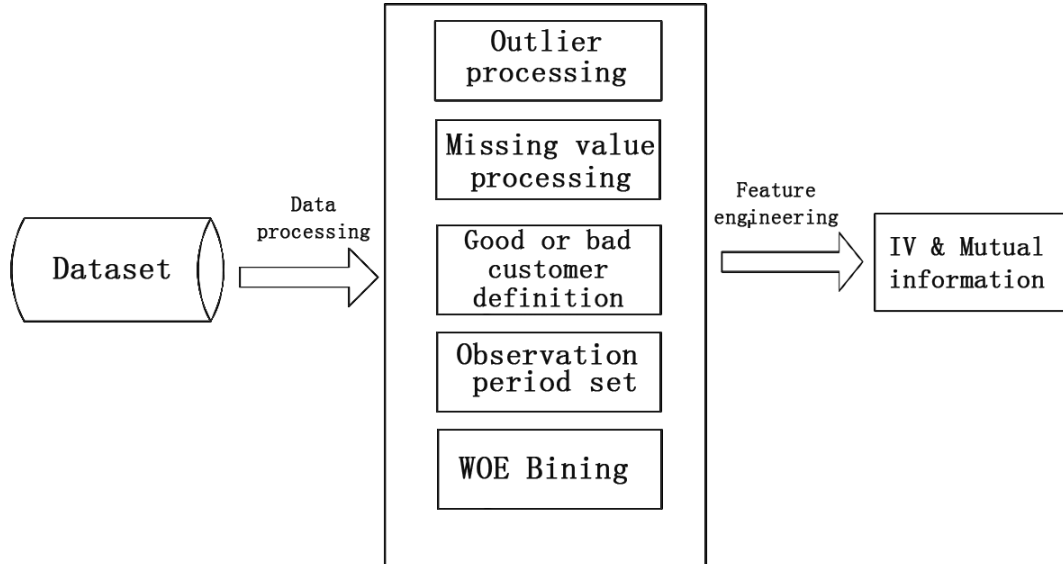


**Figure 3.4: Data processing procedure**

In this study, three advanced machine learning algorithms—XGBoost, LightGBM, and CatBoost—were evaluated as potential replacements for the traditional risk control model algorithm, logistic regression. These algorithms were selected based on their relatively high accuracy. A series of experimental tests and comparative analyses were conducted to assess their performance. Among the evaluated algorithms, CatBoost demonstrated the highest overall performance and was therefore chosen to replace the logistic regression algorithm in the risk control model.

Part 2 is the implementation process of the algorithm: According to Part 1, Catboost algorithm with better comprehensive performance (High accuracy, good classification performance, simple model algorithm (easy to use), and good interpretability, etc.) is selected to optimize the comprehensive performance of the risk control model. However,

compared with logistic regression algorithm, Catboost algorithm itself also has three major drawbacks. First, it requires a large amount of memory to process characteristic variables, thus affecting the running time of the algorithm. Second disadvantage is that CatBoost requires manual parameter adjustment, which increases the complexity of the algorithm; At the same time, the good and bad parameters of the algorithm directly affect its accuracy. The third disadvantage is that the explainability of the algorithm needs to be further improved.

Against the first disadvantage, this study chose two methods, IV (Information Value) and Mutual information, to screen the features of the dataset and delete the features with low correlation degree, which can improve the classification performance of the algorithm, which will be introduced in chapter 5.

Against the second disadvantage, in this study, GWO algorithm is selected, which has strong convergence performance, good capability in parameter seeking (Hu et al., 2021), and is able to optimize the shortcomings of CatBoost algorithm. However, GWO algorithm itself also has defects and needs to be further optimized. The disadvantage of the algorithm is it is easy to fall into local optimization and further improvement is needed. Therefore, Chapter 6 will introduce the improvement process of GWO algorithm mathematical formula in detail, and write the python core pseudo-code of the new algorithm. Develop a new algorithm, EBGWO-CatBoost, by combining the Enhanced Binary Grey Wolf Optimization (EBGWO) algorithm with the CatBoost algorithm. This hybrid approach addresses the limitation of the CatBoost algorithm, which does not automatically adjust parameters. The detailed implementation process and the complete code for this new algorithm are provided. Finally, simulation experiments were conducted for comparative verification, and the two datasets in section 3.2.1 were used.

Against the third disadvantage, this study chose the SHAP framework to enhance the explainability of CatBoost, which will be introduced in Chapter 7.

### **3.2.3 Model Validation**

In Phase 3, Algorithm Validation mainly describes the evaluation of the algorithm proposed in the previous phase. All experiments need algorithm verification and evaluation, so the third stage will run through Chapter 4, Chapter 5, Chapter 6. and Chapter 7. Meanwhile, international datasets will be used as experimental verification and comparison in these three chapters to ensure the stability, universality, and other performance characteristics of the newly proposed algorithm.

The evaluation index of risk control should not only consider the accuracy, but also consider other indicators such as the distinction between good and bad customers, because the dataset is unbalanced data. When accuracy serves as the model's performance metric, it tends to be inherently high due to the significant imbalance between the number of good and bad customer samples. Then the ability to predict bad customers is lost, therefore, it is unreasonable to use accuracy as an evaluation index in the risk control model. Therefore, the confusion matrix is used in this study because it can well avoid such problems (Marom et al., 2010). Among them, good customers are positive samples and poor customers are negative samples. The confusion matrix is defined in Figure 3.5.

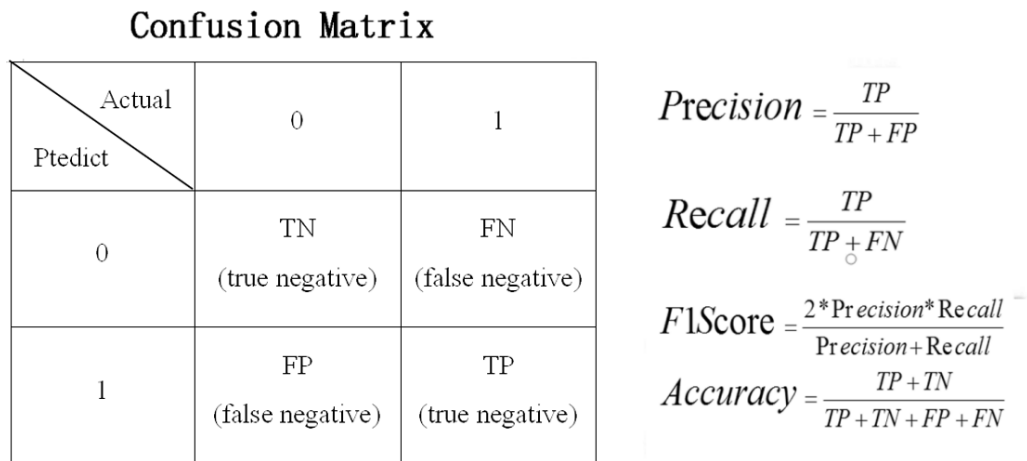
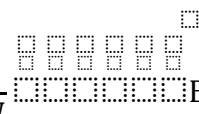


**Figure 3.5: Confusion Matrix**

Accuracy represents the percentage of all samples that the model correctly predicts. This includes both positive predicted examples that are truly positive and negative predicted examples that are truly negative.

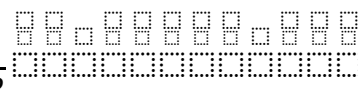
$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$



Equation 3.1

Precision is the ratio of true positive predictions to the total number of positive predictions made by the model.

$$Precision = \frac{TP}{TP + FP}$$



Equation 3.2

Recall, also known as sensitivity, is the proportion of correctly predicted positive samples to all actual positive samples.

$$Recall = \frac{TP}{TP + FN}$$



Equation 3.3

The recall is the proportion of predicted positive samples to actual positive samples.

Sometimes there will be conflicts between the accuracy rate and recall rate. In this case, these two indicators should be comprehensively evaluated and weighted, and averaged, namely F-score. When  $\alpha = 1$ , it is the most commonly used F1 value:

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad \text{Equation 3.4}$$

The Receiver Operating Characteristic (ROC) curve is a graphical plot that illustrates the performance of a classification model across different threshold settings. It displays the trade-off between the True Positive Rate (TPR) and the False Positive Rate (FPR) as the classification threshold varies. Typically, the ROC curve plots the TPR against the FPR, where the TPR is plotted on the vertical axis and the FPR is plotted on the horizontal axis. FPR signifies the proportion of negative cases incorrectly classified as positive, and its magnitude indicates the number of falsely identified positive cases within the sample. In contrast, the True Positive Rate (TPR), often referred to as sensitivity, measures the proportion of actual positive cases that are correctly identified by the model. Higher TPR values indicate a greater number of correctly predicted positive samples. In summary, the ROC curve visually illustrates the relationship between TPR and FPR, enabling the assessment of a binary classification model's performance across different thresholds. Higher TPR and lower FPR are desirable, indicating better model performance in correctly classifying positive cases while minimizing false positives.

$$TPR = \frac{TP}{TP + FN} \quad \text{Equation 3.5}$$

$$FPR = \frac{FP}{TN + FP} \quad \text{Equation 3.6}$$

The Area Under the Curve (AUC) represents the total area beneath the Receiver Operating Characteristic (ROC) curve. A greater AUC suggests a more effective predictive performance of the model. This relationship is illustrated in Figure 3.6.

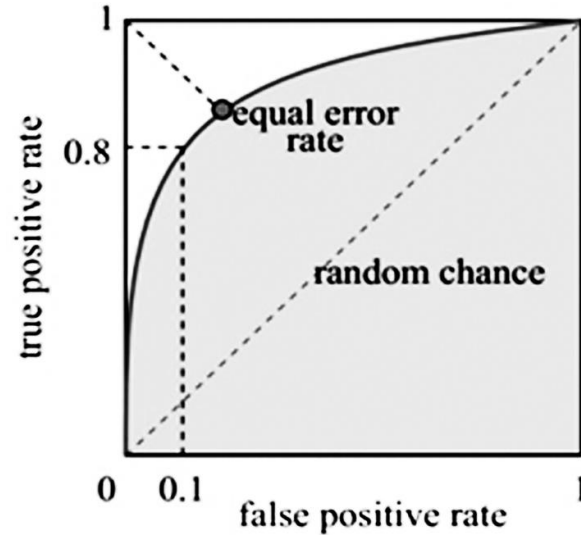


**Figure 3.6: AUC structure diagram**

### **3.3 Chapter Summary**

In this chapter, the research method of the thesis is introduced in detail. There are mainly three stages. The first stage is the definition of the problem and the introduction of the dataset, the main content is in Chapter 1 and Chapter 2. The second stage is the selection and optimization process of the algorithm, which is mainly discussed in Chapters 4, 5, 6, and 7. The third stage is to explain the evaluation method of the algorithm, the main content is in the process of all experiments in this study.

## CHAPTER 4

### ALGORITHM CHOICES

#### 4.1 Overview

This chapter provides a comprehensive comparison between traditional logistic regression and modern gradient boosting algorithms — CatBoost, LightGBM, and XGBoost — within the context of risk control models. It highlights the advantages of these newer algorithms in terms of accuracy, ability to handle non-linear data, and performance on large datasets. CatBoost, in particular, is noted for its high accuracy and effective handling of categorical data, though CatBoost has higher running times compared to logistic regression. The analysis shows that while CatBoost outperforms logistic regression in accuracy and customer distinction, CatBoost still needs improvements in computational classification performance and explainability. The study concludes that CatBoost is the superior choice over logistic regression but requires further optimization to fully replace traditional methods.

#### 4.2 Algorithm choices

The traditional logistic regression algorithm has the advantages of fast training speed, easy understanding, and good model explainability (Wang et al., 2020). It is suitable for solving linear problems, but its shortcomings are also obvious, such as low accuracy, unable to deal with nonlinear data, low accuracy, and poor overall comprehensive performance. It is a popular view that Boosting algorithm has better overall performance than logistic regression in machine learning (Liang et al., 2020). Many scholars now use XGBoost and LightGBM algorithms for data testing, and the results are significantly better than the traditional logistic regression algorithm (Zhou et al., 2019). A few scholars combined the CatBoost algorithm with XGBoost algorithm and attained some results (Wu et al., 2022).

The comprehensive performance of CatBoost, LightGBM and XGBoost algorithms in the risk control model is better than that of the traditional risk control model algorithm (Kinnander, 2020).

**Gradient Boosting Framework.** Gradient Boosting is a powerful ensemble learning technique that builds models sequentially, where each new model attempts to correct the errors of its predecessor. This process helps in capturing complex patterns and interactions within the data. Here's how each algorithm leverages this framework: 1. XGBoost: Stands for eXtreme Gradient Boosting. It improves traditional gradient boosting by optimizing both the model's performance and computational classification performance. Key features include: Regularization: XGBoost includes L1 (Lasso) and L2 (Ridge) regularization to reduce overfitting. Tree Pruning: XGBoost uses a more efficient method for tree pruning which allows it to handle complex data structures better. Parallel Processing: It supports parallelization of tree construction, making it faster. 2. LightGBM: Light Gradient Boosting Machine is optimized for speed and classification performance. It's particularly effective on large datasets and provides: Histogram-based Learning: LightGBM uses histogram-based algorithms to bin continuous features, which speeds up training and reduces memory consumption. Leaf-wise Growth: Instead of level-wise growth like in XGBoost, LightGBM grows trees leaf-wise, which often results in better accuracy. 3. CatBoost: Categorical Boosting is designed to handle categorical features effectively without the need for extensive preprocessing. It includes: Ordered Boosting: CatBoost uses an ordered boosting algorithm that helps to reduce overfitting and bias in the training process. Efficient Handling of Categorical Features: CatBoost can work with categorical features directly, using techniques like target encoding which helps in capturing the importance of these features.

Handling Categorical Data. Traditional models often require extensive preprocessing of categorical variables, such as one-hot encoding or label encoding. Modern algorithms like CatBoost handle categorical features natively, which can:

- Reduce Preprocessing Time: Eliminates the need for manual encoding and can handle high-cardinality features efficiently.
- Improve Model Accuracy: More effectively capture the relationship between categorical features and the target variable.

Regularization and Overfitting Control. Overfitting is a common challenge in machine learning models, where a model performs well on training data but poorly on unseen data. Modern gradient boosting algorithms incorporate advanced regularization techniques to combat overfitting:

- XGBoost: Offers L1 and L2 regularization to prevent overfitting.
- LightGBM: Uses techniques such as the minimum data in a leaf and maximum depth to control model complexity.
- CatBoost: Employs ordered boosting and regularization methods to ensure robust performance and avoid overfitting.

Computational Classification performance.Speed: LightGBM and CatBoost are known for their fast training times due to efficient implementations and support for parallel processing. LightGBM's histogram-based approach is particularly quick with large datasets. Scalability: These algorithms can handle large datasets and complex models more efficiently than traditional algorithms, which often struggle with scalability. Robustness and Flexibility. Handling Missing Values: Modern algorithms have built-in mechanisms to handle missing values, reducing the need for imputation. Parameter Tuning: Advanced algorithms offer extensive hyperparameter tuning options that can lead to better model performance when appropriately tuned. Practical Performance. In practical applications, especially in risk control models which often involve complex and high-dimensional data, these modern algorithms tend to:

- Provide Higher Accuracy: Due to their sophisticated boosting methods and ability to handle

complex feature interactions. Improve Model Interpretability: Tools and libraries for these algorithms often provide built-in features for model interpretation and evaluation.

Conclusion: Modern gradient boosting algorithms like CatBoost, LightGBM, and XGBoost offer significant improvements over traditional risk control model algorithms due to their advanced techniques in handling categorical data, regularization, computational classification performance, and overall robustness. Their ability to model complex relationships and scale effectively makes them superior choices for tasks such as risk control, where accuracy and classification performance are critical.

According to the above studies of experts and scholars, these three algorithms (CatBoost, LightGBM and XGBoost) are compared with the algorithms of the traditional risk control model (logistic regression), as shown in Table 4.1.

**Table 4.1: Comparison of algorithm characteristics**

| Characteristics | LightGBM                                                | CatBoost                                                                                        | XGBoost                                         | logistic regression                                                                     |
|-----------------|---------------------------------------------------------|-------------------------------------------------------------------------------------------------|-------------------------------------------------|-----------------------------------------------------------------------------------------|
| Advantages      | High speed; less high speed; less resource consumption. | High accuracy; Speed and explainability are between LightGBM and XGBoost.                       | High accuracy; Good explainability.             | High speed; Simple algorithm; good explainability.                                      |
| Disadvantages   | Lower accuracy than XGboost; Weak explainability.       | Speed, and explainability are between LightGBM and XGBoost; Adjusting parameters is cumbersome. | Slow speed; Adjusting parameters is cumbersome. | Accuracy is low, and the ability to distinguish between good and bad customers is weak. |

Experts and scholars (Yu et al., 2024) conducted data testing on Logistic regression, LightGBM, XGBoost, and Catboost algorithms. The test results show that the accuracy of the logistic regression algorithm is more than 10% lower than that of LightGBM, XGBoost,

and CatBoost. The computation time of logistic regression is similar to that of LightGBM, 30% faster than CatBoost algorithm, and about 50% faster than XGBoost algorithm. Compared the accuracy of LightGBM, XGBoost, and Catboost algorithms, the accuracy of CatBoost algorithm is about 5% faster than LightGBM and 2% lower than XGBoost algorithm. LightGBM has the fastest computing speed, about 20% faster than CatBoost and about 60% faster than XGBoost. The overall situation is as follows: Logistic regression's advantages: fast training speed, Simple algorithm, good explainability; Disadvantages: accuracy is low, and the ability to distinguish between good and bad customers is weak; LightGBM: Fast, low resource consumption, but lower accuracy than Xgboost; XGBoost's advantages: high accuracy. Cons: Slow; CatBoost: has the advantages of both LightGBM and XGBoost, but it needs to tune "super parameters" to get high accuracy. The algorithm with the highest comprehensive performance is selected from the three algorithms to replace the logistic regression algorithm in the risk control model.

Following this, perform experimental tests and comparative analysis using two different datasets. The process of the experiment is shown in Figure 4.1.

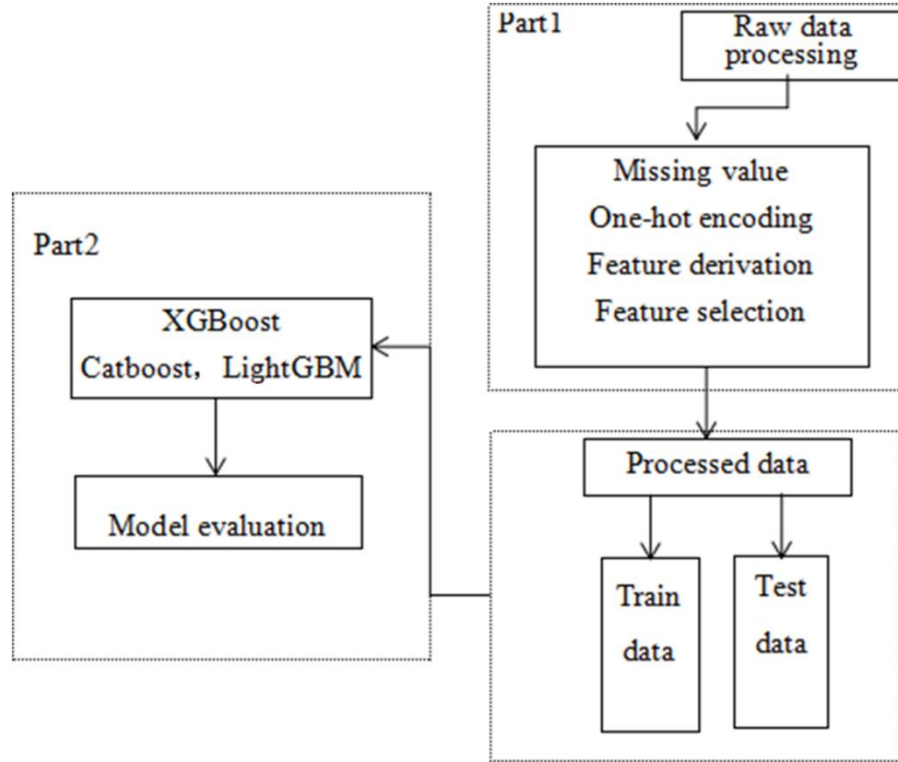


**Figure 4.1: Experimental flow structure diagram**

Part1 is the data processing part, which is about dataset processing (data preprocessing, see Section 3.2.2). Among part1, feature standardization: The main functions of feature standardization include eliminating the dimensional differences between different variables, improving the rationality of model training, speeding up model fitting, and ensuring that each dimensional feature has the same weight in the model's objective function. It is primarily reflected in improving the model's accuracy and reliability, ensuring that different features are fairly evaluated in the model, and thereby enhancing the model's generalization ability. Feature selection: Feature selection involves using the results of IV (Information Value) and mutual information calculations to remove features with low relevance, which is also known as dimensionality reduction. The data is then split into 70% training data and 30% test data. Part2 is to put 70% of the data into the risk control model for training the model, and use 30% of the data for testing. Then, the default parameters of the three algorithms are

used, and the main parameters of the three algorithms are shown in Tables 4.2 - 4.4. Finally, the algorithm was tested and evaluated.

**Table 4.2: CatBoost important parameters**

| Model Parameters       | Parameters meaning                                                                                                                                                | Value    |
|------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------|
| Loss_function          | Loss function, supported by RMSE, Logloss, MAE Quantile, LogLin Quantile, Poisso                                                                                  | Logloss  |
| Iterations             | Maximum number of trees.                                                                                                                                          | 100      |
| Learning_rate          | Learning rate determines how fast or slow the model will learn.                                                                                                   | 0.03     |
| L2_leaf_reg            | Regular parameter.                                                                                                                                                | 3        |
| Depth                  | The depth of the tree, maximum 16, recommended between 1 and 10.                                                                                                  | 6        |
| Use_best_model         | When setting this parameter, you need to provide test data. The number of trees can be obtained by training parameters and optimizing loss function.              | False    |
| Best_model_min_trees   | Minimum number of trees that the best model should have.                                                                                                          | False    |
| Leaf_estimation_method | Method of calculating leaf value, Newton/Gradient.                                                                                                                | Gradient |
| Boosting_type          | The lifting scheme can be used for the classical gradient lifting scheme, and can also be used for ordered, which can provide better quality on smaller datasets. | Ordered  |
| Eval_metric            | Lossfunction for over-fitting inspection (set True) and best model selection (set True)                                                                           | True     |
| Random_seed            | Random number seed.                                                                                                                                               | 0        |
| Custom_metric          | Customize the evaluation indicators output during training.                                                                                                       | None     |
| Feature_weights        | information gain of each feature is calculated when the subtree is split                                                                                          | 1        |

**Table 4.3: LightGBM important parameters**

| Model Parameters  | Parameters meaning                                               | Value  |
|-------------------|------------------------------------------------------------------|--------|
| Boosting_typ      | For gradient-boosted decision tree-based methods.                | gbdt   |
| Max_depth         | Determine the depth of the number                                | -1     |
| Num_leaves        | Controlling the number of leaf nodes in each decision tree       | 31     |
| Min_child_samples | Minimum number of leaf node samples                              | 20     |
| Minchild_weight   | Synthesis of all sample weights at the smallest leaf node        | 1      |
| Feature_fraction  | Control how many features are selected each time the tree grows  | 0.5    |
| Bagging_fraction  | Controlling how much data is used in each iteration of the model | 0.5    |
| Bagging_freq      | Tell the model to consider subsampling every niterations         | 0      |
| objective         | It depends on the specific problem                               | binary |

**Table 4.4: XGBoost important parameters**

| Model Parameters | Parameters meaning                                                                      | Value        |
|------------------|-----------------------------------------------------------------------------------------|--------------|
| Booster          | Two parameter options: gotree (tree structure to run the data) and gblinear             | gbtree       |
| Subsample        | Proportion of subsamples to the training set                                            | 1            |
| Learning_rate    | Specifying the learning rate                                                            | 0.3          |
| N_estimator      | The system default is 0.3                                                               | 0.3          |
| Gamma            | The system defaults to 0, and the larger the value, the more conservative the algorithm | 0            |
| Min_child_weight | Minimum number of samples on a leaf                                                     | [1, 3, 5, 7] |
| Max_depth        | Specifies the maximum depth of the tree                                                 | 6            |
| Colsample_bytree | Column sampling ratio                                                                   | 1            |

#### 4.2.1 Experimental results of China\_Credit dataset

As mentioned in the previous section, all the 20 characteristic variables of China\_Credit dataset were put into the model for testing after data standardization. The details of the 20 characteristic variables are shown in Table 4.5. Then the parameters used by the three algorithms (CatBoost, LightGBM and XGBoost) are in Table 4.1-4.3 respectively. At the same time, the data of 20 characteristics are also put into the traditional risk control model (logistic regression algorithm) for the experiment.

**Table 4.5: The variables of China\_Credit dataset**

| Attribute Name                    | Datatype | Value distribution                                                                                                                                                                                                                        |
|-----------------------------------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Account Balance                   | Int      | Account balance                                                                                                                                                                                                                           |
| Duration of Credit (month)        | Int      | Duration in month                                                                                                                                                                                                                         |
| Payment Status of Previous Credit | Int      | Value:0-4, (0:no credits taken/ all credits paid back duly, 1 : all credits at this bank paid back duly, 2 : existing credits paid back duly till now, 3 : delay in paying off in the past, 4 : critical account/ other credits existing) |
| Purpose                           | Int      | Value:1-10, (0:newcar、 1 : used car、 2 : furniture/equipment、 3 : radio/television、 4 : domestic appliances、 5 : repairs、 6 : education、 7 :vacation - does not exist、 8 : retraining、 9 : business、 10 : others)                         |
| Credit Amount                     | Int      | Credit amount                                                                                                                                                                                                                             |
| Value Savings/Stocks              | Int      | Value:1-5, (1: $X < 100$ , 2: $100 \leq X < 500$ , 3: $500 \leq X < 1000$ , 4: $X \geq 1000$ , 5: unknown/ no savings account)                                                                                                            |

|                               |         |                                                                                                                                                                                   |
|-------------------------------|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Length of current employment  | Int     | Length of service                                                                                                                                                                 |
| Instalment per cent           | Int     | Installment rate in percentage of disposable income                                                                                                                               |
| Sex & Marital Status          | Int     | Value:1-5, (1 : male : divorced/separated、 2 : female : divorced/separated/married, 3 : male : single, 4 : male : married/widowed、 5 : female : single)                           |
| Guarantors                    | Int     | Value: 1-3,, (1 : none, 2 : co-applicant, 3 : guarantor)                                                                                                                          |
| Duration in Current address   | Int     | Present residence since                                                                                                                                                           |
| Most valuable available asset | Int     | Value:1-4,(1: real estate,2: if not “1” - building society savings agreement/ life insurance、 3 : if not “1” / “1” 2- car or other, not in attribute 、 4 : unknown / no property) |
| Age (years)                   | Int     | Age in years                                                                                                                                                                      |
| Concurrent Credits            | Int     | Value:1(bank), 2(stores),3 (none)                                                                                                                                                 |
| Type of apartment             | Int     | Value:1(rent), 2 (own),3 (for free)                                                                                                                                               |
| No of Credits at this Bank    | Int     | Number of existing credits at this bank                                                                                                                                           |
| Occupation                    | Int     | Value:1(unemployed/ unskilled - non-resident), 2(unskilled - resident) 3 (skilled employee / office) 4(management/ self-employed/highly qualified employee/ officer)              |
| No of dependents              | Int     | Number of people being liable to provide maintenance for                                                                                                                          |
| Telephone                     | Boolean | registered under the customers’ name, value: truth or false                                                                                                                       |
| Foreign Worker                | Boolean | whether foreign employees, value: false or truth                                                                                                                                  |

To more clearly demonstrate the differences in running time among the four algorithms, the memory usage, CPU usage, and other hardware resources were fixed at a constant value during the experiments. Then the test is carried out, and the results are shown in Figure 4.2.

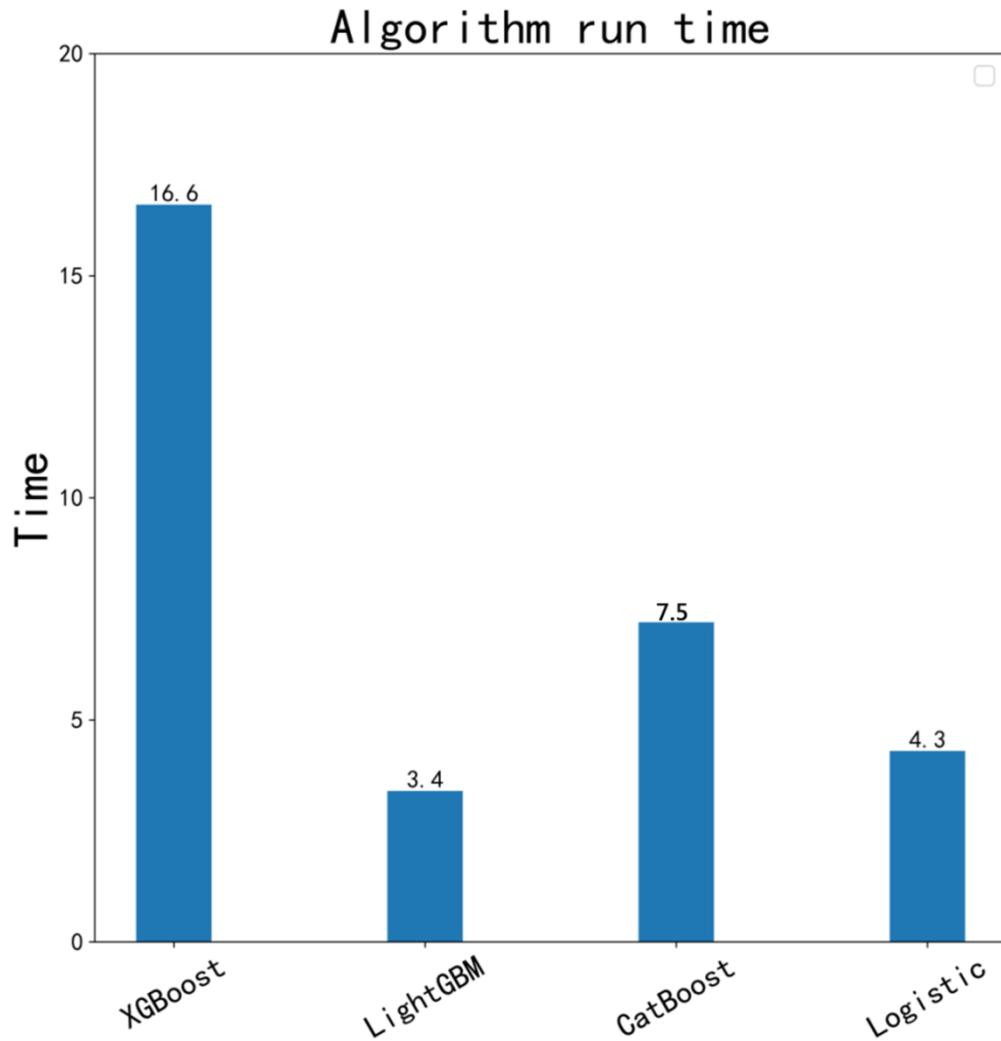


**Figure 4.2: Algorithm run time (20 variables)**

The Comprehensive performance of three algorithms (CatBoost, LightGBM and XGBoost) and logistic regression algorithm are shown in Table 4.6. In the table, Recall, AUC, F1 are indicators of the ability to distinguish good and bad customers.

**Table 4.6: Performance indicators of different algorithm (unit: %)**

| Algorithm name | Accuracy | Precision | Recall | AUC   | F1    | Number features |
|----------------|----------|-----------|--------|-------|-------|-----------------|
| CatBoost       | 82.66    | 81.06     | 82.92  | 82.51 | 81.98 | 20              |

|          |       |       |       |       |       |    |
|----------|-------|-------|-------|-------|-------|----|
| XGBoost  | 81.92 | 80.27 | 80.31 | 79.84 | 80.07 | 20 |
| LightGM  | 73.49 | 74.64 | 72.94 | 77.72 | 73.78 | 20 |
| Logistic | 70.33 | 71.01 | 71.12 | 73.43 | 71.06 | 20 |

---

#### 4.2.2 Experimental analysis of China\_Credit dataset

As can be seen from Figure 4.2, all feature variables are put into the model for testing, the running time of the three algorithms (CatBoost, LightGBM, and XGBoost) is as follows: when the data source sample is more than 10K, the XGBoost algorithm takes an average of 16.6 seconds, the CatBoost algorithm 7.5 seconds, and LightGMB 3.4 seconds. From the analysis of the running time of the algorithm, the running time of the XGBoost algorithm is more than 10 seconds different from that of the logistic regression algorithm. Therefore, from the perspective of the running time of the algorithm, XGBoost is not suitable to replace the logistic regression algorithm. The time difference between the running time of the LightGMB algorithm and the logistic regression algorithm is about 1 second, and the difference between the CatBoost algorithm and the logistic regression algorithm is about 3.2 seconds. If other factors such as accuracy are not considered, it is more appropriate to choose the LightGMB algorithm.

It can be seen from Table 4.6 that "Accuracy and Precision" are the Accuracy indicators of the risk control model. Among them, the average accuracy of "accuracy" of CatBoost is 82.66%, that of XGBoost is 81.92%, and that of LightGMB is 73.49%. The CatBoost algorithm differs little from the XGBoost algorithm, while LightGMB differs from CatBoost and XGBoost by about 10%. The results of Recall, AUC (Area Under Curve), and F1 indicators are also analyzed in terms of their ability to distinguish between good and bad customers. It can be seen from Table 4.6 that CatBoost has the best ability to distinguish,

followed by XGBoost, and LightGMB has the worst ability. So in terms of model accuracy and ability to distinguish between good and bad customers, CatBoost and XGBoost are good choices.

To sum up, the first question raised from Section 1.4 of Chapter 1 is to improve the accuracy of the model and the ability of the model to distinguish well. It is better to choose XGBoost and CatBoost algorithm, but the running time of XGBoost algorithm is too slow, with a difference of more than 10 seconds compared with logistic regression. Therefore, only CatBoost algorithm is the best choice.

However, CatBoost algorithm takes 40% higher time than logistic regression algorithm, so it needs further improvement in calculating the running time. This part of the experiment will be described in Section 5.2.

### 4.2.3 Experimental results of LendingClub dataset

In order to test the stability and generality of CatBoost, this section uses the second data, LendingClub, as the most comparative test. Same as the experimental method of China\_Credit dataset in Section 4.2.1, all 110 characteristic variables of LendingClub dataset were put into the model for testing. The 110 characteristic variables is shown in Table 4.7, and the details are shown in Appendix C. Then three algorithms (CatBoost, LightGBM, and XGBoost) use the same parameters in the previous experiments. At the same time, the data of 110 characteristics are also put into the traditional risk control model (logistic regression algorithm) for the experiment.

**Table 4.7: Cases of feature variables for part of the LendingClub dataset**

| Number | Attribute Name | Datatype | Value distribution |
|--------|----------------|----------|--------------------|
|--------|----------------|----------|--------------------|

|     |                  |     |                                                                                           |
|-----|------------------|-----|-------------------------------------------------------------------------------------------|
| 1   | installment      | Int | The number of months of repayment, Value:1-360.                                           |
| 2   | total_rec_prncp  | Int | Principal received to date                                                                |
| 3   | last_pymnt_amnt  | Int | The total amount of the last payment received                                             |
| 4   | total_pymnt      | Int | The total amount of payments received so far                                              |
| 5   | total_pymnt_inv  | Int | Part of the total amount of money funded by investors received to date                    |
| 6   | int_rate         | Int | Indicates whether income is LC verified, not verified, or whether the source of income is |
| 7   | total_rec_int    | Int | Interest received to date                                                                 |
| ... | ...              | Int | ...                                                                                       |
| 107 | open_acc_6m      | Int | The number of open trades in the past 6 months                                            |
| 108 | all_util         | Int | All transactions equalize credit limits                                                   |
| 109 | pub_rec          | Int | Devalue the number of public records                                                      |
| 110 | application_type | Int | Type of Application                                                                       |

In the experiment, the operating memory, CPU usage and other hardware condition resources of the three algorithms are set as in section 4.2.1. The results are Figure 4.3.

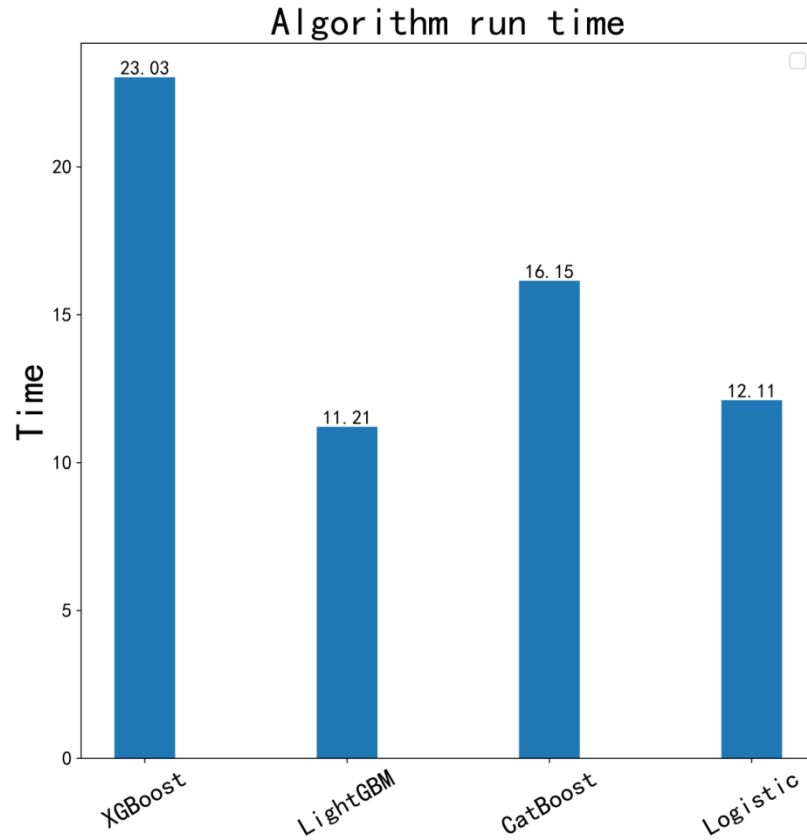


**Figure 4.3: Algorithm run time (110 variables)**

The Comprehensive performance of three algorithms (CatBoost, LightGBM and XGBoost) and logistic regression algorithm are shown in Table 4.8. In the table, Recall, AUC, F1 are indicators of the ability to distinguish good and bad customers.

**Table 4.8: Performance indicators of different algorithm (unit: %)**

| Algorithm name | Accuracy     | Precision    | Recall      | AUC          | F1           | Number features |
|----------------|--------------|--------------|-------------|--------------|--------------|-----------------|
| CatBoost       | <b>89.97</b> | <b>88.66</b> | <b>86.1</b> | <b>87.09</b> | <b>87.37</b> | 110             |
| XGBoost        | 85.50        | 84.31        | 83.7        | 83.87        | 84.05        | 110             |
| LightGM        | 82.51        | 76.55        | 80.1        | 83.55        | 78.29        | 110             |
| Logistic       | 73.33        | 71.12        | 72.3        | 74.80        | 71.72        | 110             |

#### 4.2.4 Experimental analysis of LendingClub dataset

As can be seen from the analysis in Figure 4.3, the time consumed by XGBoost algorithm is very long, which takes 23.03 seconds, CatBoost algorithm 16.15 seconds, LightGMB 11.21 seconds, and logistic regression algorithm 12.11 seconds. The classification performance of XGBoost algorithm is too low, so CatBoost and LightGMB algorithm can only be selected.

It can be seen from Table 4.8 that "Accuracy and Precision" are the Accuracy indicators of the risk control model. Among them, the average accuracy of "accuracy" of CatBoost is 89.97%, that of XGBoost is 85.50%, and that of LightGMB is 73.49%. The CatBoost algorithm differs little from the XGBoost algorithm, while LightGMB differs from CatBoost and XGBoost by about 13%. The CatBoost algorithm has an accuracy improvement of about 20% compared to the logistic regression algorithm. The results of Recall, AUC (Area Under Curve), and F1 indicators are also analyzed in terms of their ability to distinguish between good and bad customers. It can be seen from Table 4.8 that CatBoost has the best ability to distinguish, followed by XGBoost, and LightGMB has the worst ability. So in terms of model accuracy and ability to distinguish between good and bad customers, CatBoost and XGBoost are good choices.

From the above analysis, it can be seen that the CatBoost algorithm combines the advantages of LightGBM and XGBoost, The CatBoost algorithm has a high accuracy rate and good ability to distinguish between good and bad customers, and the algorithm running time is also not bad. The running time of this algorithm is also about 30% higher than that of the logistic regression algorithm, which needs to be further optimized. This is also similar to the results for the China\_Credit dataset.

### **4.3 Compare of CatBoost and Logistic regression Algorithms**

As mentioned in Chapter 1, Section 1.3, the traditional risk control model uses a logistic regression algorithm. Its main advantages are high classification performance, simple use (low algorithm complexity), and strong explainability. However, its accuracy is low, the ability to distinguish between good and bad customers is poor, and it needs to be optimized. Through the experiments in sections 4.2 - 4.3, the CatBoost algorithm is selected to replace the algorithm of the traditional risk control model, because it has high accuracy and ability to distinguish good from bad customers. But compared with the logistic regression algorithm, the CatBoost algorithm not only has to solve this problem but also has the main advantages of the logistic regression algorithm. Therefore, CatBoost algorithm also needs to solve three problems: 1. High classification performance (algorithm running time), 2. Low algorithm complexity, 3. strong explainability. As can be seen from Figure 4.2 - 4.3, the running time of the CatBoost algorithm needs to be further improved, which will be introduced in Chapter 5. In addition, if the CatBoost algorithm wants to obtain a higher accuracy rate, it must be manually adjusted (Li et al., 2020), CatBoost has a large number of parameters, and manual adjustment of parameters will greatly increase the complexity of the algorithm. Compared with the logistic regression algorithm, the complexity of the CatBoost algorithm needs to be further simplified, which will be introduced in Chapter 6. 3. The explainability of the CatBoost algorithm needs to be further improved (Zhao, 2023) compared with the logistic regression algorithm.

### **4.4 Conclusion**

From the analysis of the above experimental results, it can be seen that CatBoost algorithm has the highest accuracy and customer identification ability among the three algorithms (LightGBM, XGBoost, CatBoost), and can solve the first problem proposed in

section 1.4, that is, improve the accuracy of the risk control model in identifying good and bad customers (It has improved by about 20% compared to logistic regression algorithms). However, the classification performance of the algorithm (algorithm running time) still needs to be improved. As can be seen from the analysis in section 4.3, CatBoost algorithm needs to solve the following problems in order to completely replace logistic regression algorithm: 1. Classification performance of the algorithm; 2. Algorithm complexity; 3. The explainability of the algorithm. The next chapter addresses the "algorithmic classification performance" problem.

#### **4.5 Chapter Summary**

This chapter introduces the content of Phase 2 Algorithm Choices in Chapter 3. Through the experimental analysis of CatBoost, LightGBM and XGBoost algorithms, The CatBoost algorithm with the highest comprehensive performance is selected to replace the logistic regression algorithm of the traditional risk control model. Then compared with logistic regression algorithm, find out the three main defects of CatBoost algorithm, and prepare for the following.

## CHAPTER 5

### CATBOOST CLASSIFICATION OPTIMIZATION

#### 5.1 Overview

In view of the first question problems raised in section 1.4, which is the model has high accuracy and good and bad customer identification ability, at the same time it has high classification performance. From the experimental results in Chapter 4, it can be seen that the CatBoost algorithm solves this problem with high accuracy and the ability to identify good and bad customers. However, the problem of algorithmic classification performance has not been solved. This chapter addresses this issue further. The main contents are summarized as follows:

5.1.1 China\_Credit Dataset. For the China\_Credit dataset, we removed 7 features with low IV values. The impact of this reduction on model performance was analyzed. Post-dimension reduction, we observed that the algorithm running times improved. The CatBoost algorithm showed a significant reduction in running time and performed similarly to logistic regression in terms of speed, but with better accuracy and customer distinction.

5.1.2 LendingClub Dataset. The LendingClub dataset was used to validate the findings. We applied similar dimensionality reduction, removing 11 features with low IV values. The experimental results demonstrated that while accuracy and customer differentiation slightly decreased with fewer features, the CatBoost algorithm maintained strong performance with improved running times. The CatBoost algorithm's performance was consistent with that observed in the China\_Credit dataset, showing its effectiveness in balancing accuracy and computational classification performance.

5.1.3 Summary. Dimensionality reduction enhanced the algorithm performance by improving running times without significantly affecting accuracy. CatBoost emerged as the superior algorithm, offering better classification performance and accuracy compared to logistic regression and other gradient boosting methods. Future work will focus on optimizing CatBoost further and exploring automatic parameter tuning to address algorithmic complexity.

## 5.2 Features selection and algorithm test

The two datasets used in this chapter are the same as in Chapter 4, and then dimensionality reduction is performed on the dataset, that is, the number of characteristic variables is reduced and the variables with low correlation are removed. The main process is as follows: First, IV is used to compute the values of each feature of the dataset, and the variable features whose IV is less than 0.02 are deleted (see 2.6.3). Secondly, Mutual information (Shannon, 1948) is used to calculate the feature correlation value again for verification, and the feature variable with the highest correlation degree is determined. Finally, the same algorithm parameters as Tables 4.1 - 4.3 were used for the experiment.

### 5.2.1 Experimental results of China\_Credit dataset

The calculation method for feature IV values is detailed in Section 2.6.3. The IV and mutual information calculation results for the features are presented in Table 5.1.

**Table 5.1: IV and mutual information values of China\_Credit dataset**

| Feature Name                | IV value              | Mutual value          | Whether high relevance |
|-----------------------------|-----------------------|-----------------------|------------------------|
| No of dependents            | $4.34 \times 10^{-5}$ | $4.55 \times 10^{-6}$ | NO                     |
| Duration in Current address | $2.4 \times 10^{-4}$  | $3.80 \times 10^{-4}$ | NO                     |

|                               |                       |                       |     |
|-------------------------------|-----------------------|-----------------------|-----|
| Telephone                     | $6.38 \times 10^{-3}$ | $6.70 \times 10^{-4}$ | NO  |
| Occupation                    | $8.10 \times 10^{-3}$ | $9.30 \times 10^{-4}$ | NO  |
| Age (years)                   | $1.10 \times 10^{-2}$ | $3.20 \times 10^{-3}$ | NO  |
| No of Credits at this Bank    | $1.10 \times 10^{-2}$ | $1.40 \times 10^{-3}$ | NO  |
| Guarantors                    | $1.60 \times 10^{-2}$ | $3.3 \times 10^{-3}$  | NO  |
| Purpose                       | $2.40 \times 10^{-2}$ | $1.70 \times 10^{-2}$ | Yes |
| Instalment per cent           | $2.60 \times 10^{-2}$ | 0.0038                | Yes |
| Type of apartment             | $3.90 \times 10^{-2}$ | 0.0091                | Yes |
| Sex & Marital Status          | $4.30 \times 10^{-2}$ | $4.70 \times 10^{-3}$ | Yes |
| Foreign Worker                | $4.40 \times 10^{-2}$ | $4.00 \times 10^{-3}$ | Yes |
| Concurrent Credits            | $5.80 \times 10^{-2}$ | $6.20 \times 10^{-3}$ | Yes |
| Most valuable available asset | $7.80 \times 10^{-2}$ | $1.20 \times 10^{-2}$ | Yes |
| Length of current employment  | $8.30 \times 10^{-2}$ | $9.10 \times 10^{-3}$ | Yes |
| Credit Amount                 | $9.50 \times 10^{-2}$ | $5.70 \times 10^{-1}$ | Yes |
| Value Savings/Stocks          | $1.70 \times 10^{-1}$ | $1.90 \times 10^{-2}$ | Yes |
| Duration of Credit (month)    | $2.30 \times 10^{-2}$ | $4.40 \times 10^{-2}$ | Yes |
| Payment Status of Previous    | $2.90 \times 10^{-1}$ | $4.00 \times 10^{-2}$ | Yes |
| Account Balance               | $6.30 \times 10^{-1}$ | $6.60 \times 10^{-2}$ | Yes |

It can be seen from the above Table 5.1 that there are 7 feature variables with IV values less than 0.02. According to the analysis in Section 2.6.3 of Chapter 2, IV values less than 0.02 are of low correlation with the comprehensive performance of the model, so they need to be deleted. At the same time, the mutual information values of these feature variables are

also the smallest, indicating that these features have little impact on the final accuracy of the model and can be deleted. The deleted variables are shown in Table 5.2.

**Table 5.2: Feature variables deletion**

| Feature Name               | IV value              | Mutual value          | Whether high |
|----------------------------|-----------------------|-----------------------|--------------|
| No of dependents           | $4.34 \times 10^{-5}$ | $4.55 \times 10^{-6}$ | NO           |
| Duration in Current        | $2.4 \times 10^{-4}$  | $3.8 \times 10^{-4}$  | NO           |
| Telephone                  | $6.4 \times 10^{-3}$  | $6.7 \times 10^{-4}$  | NO           |
| Occupation                 | $8.1 \times 10^{-3}$  | $9.3 \times 10^{-4}$  | NO           |
| Age (years)                | $1.1 \times 10^{-2}$  | $3.2 \times 10^{-4}$  | NO           |
| No of Credits at this Bank | $1.1 \times 10^{-2}$  | $1.4 \times 10^{-3}$  | NO           |
| Guarantors                 | $1.6 \times 10^{-2}$  | $3.3 \times 10^{-3}$  | NO           |

After deleting the variables with low correlation, the remaining 13 variables were put into the model for experiments. The content of the test, cpu usage and other hardware resource control is the same as section 4.2.1. The experimental results are as follow. Figure 4.4 shows the time used by different algorithms after optimization.

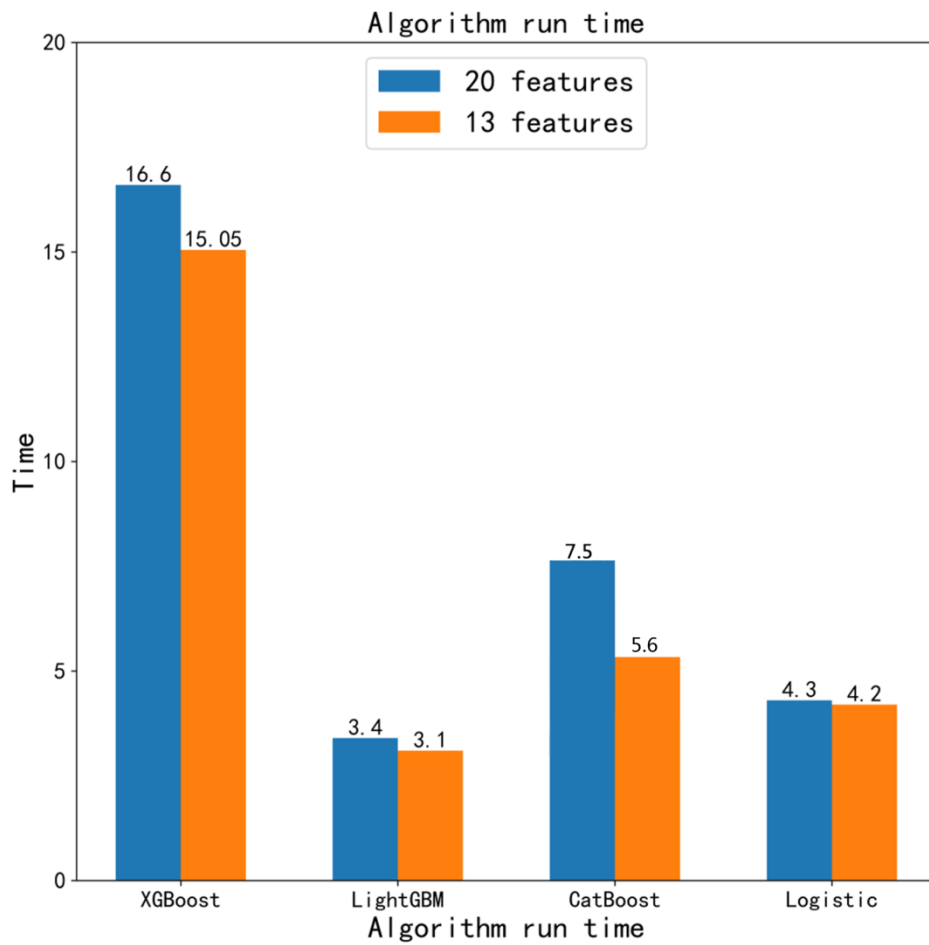


**Figure 5.1: Algorithm running time (13 variables)**

The Comprehensive performance of three algorithms (CatBoost, LightGBM and XGBoost) and logistic regression algorithm are shown in Table 5.3. In the table, Recall, AUC, F1 are indicators of the ability to distinguish good and bad customers.

**Table 5.3: Performance indicators of different algorithm (unit: %)**

| Algorithm name | Accurac      | Precisio     | Recall       | AUC          | F1           | Number features |
|----------------|--------------|--------------|--------------|--------------|--------------|-----------------|
| CatBoost       | <b>82.05</b> | <b>82.71</b> | <b>82.51</b> | <b>82.03</b> | <b>82.61</b> | <b>13</b>       |
| CatBoost       | 82.66        | 81.06        | 82.92        | 82.51        | 81.98        | 20              |
| XGBoost        | 81.44        | 79.92        | 80.02        | 79.54        | 79.97        | 13              |

|          |       |       |       |       |       |    |
|----------|-------|-------|-------|-------|-------|----|
| XGBoost  | 81.92 | 80.27 | 80.31 | 79.84 | 80.29 | 20 |
| LightGMB | 73.06 | 74.27 | 72.64 | 77.57 | 73.45 | 13 |
| LightGMB | 73.49 | 74.64 | 72.94 | 77.72 | 73.78 | 20 |
| Logistic | 70.01 | 70.98 | 70.85 | 73.01 | 70.91 | 13 |
| Logistic | 70.33 | 71.01 | 71.12 | 73.43 | 71.06 | 20 |

---

### 5.2.2 Experimental analysis of China\_Credit dataset

From the analysis of Figure 5.1, the running time of these three algorithms has a certain amount of reduction due to the reduction of 7 feature variables. XGBoost algorithm is too slow and takes more than 15.02 seconds, the LightGBM algorithm is 3.1 seconds, which is not much different from that before feature screening. The running time of CatBoost algorithm is 5.6 seconds, which is about 20% less than before feature screening. The running time difference between CatBoost and logistic regression algorithm is about 1 second, and the running time of the two algorithms is similar.

In Figure 5.1 and Table 5.3, the analysis of the algorithm execution time shows that a key criterion is that the classification speed must be fast enough. The processing time of XGBoost is significantly longer than that of CatBoost. Therefore, the speed required for effective risk control modeling cannot be met. In terms of accuracy, the accuracy of XGBoost algorithm is lower than that of CatBoost. Therefore, CatBoost algorithm performs better than XGBoost.

From Table 5.3, although the average value of "Accuracy and Precision" decreased slightly after feature variable screening (13 features), the overall value was similar to that without feature variable screening (20 features). Other indicators, such as Recall, AUC, and

F1, which distinguish good from bad customer ability, only decreased slightly, compared with before the dimensionality reduction (feature screening). Therefore, after dimensionality reduction by IV value, the overall performance of the algorithm is not significantly reduced, and the running time of the algorithm is improved, especially the CatBoost algorithm is significantly improved, thus, the classification performance of the algorithm is improved. The first drawback of CatBoost algorithm is solved to a certain extent, that is, the problem of too many variable characteristics affecting the classification performance of the algorithm. Finally, compared with the running time of the logistic regression algorithm of the traditional risk control Compared with logistic regression algorithm, CatBoost algorithm has a difference of about 1 second, the two algorithms running at similar times. But it has a better accuracy and the ability to distinguish between good and bad customers.

In summary, through feature variable screening (dimensionality reduction), the running time of the algorithm can be effectively improved without affecting the comprehensive performance of the CatBoost algorithm, thus improving the classification performance of the algorithm. To verify this conclusion, this study will use another dataset in the next section to test and verify the comparison.

### **5.2.3 Experimental results of LendingClub dataset**

In this section, the LendingClub dataset is used as a control experiment. The experimental procedures are the same as in Method 5.2.1. The calculation results of feature IV and Mutual information are shown in Table 5.4, and the details are shown in Appendix D.

**Table 5.4: IV values for some feature variables of the LendingClub dataset**

| Attribute Name      | IV value              | Mutual information     | Whether high |
|---------------------|-----------------------|------------------------|--------------|
| installment         | $9.36 \times 10^0$    | $1.88 \times 10^{-2}$  | yes          |
| total_rec_prncp     | $9.30 \times 10^0$    | $5.54 \times 10^{-2}$  | yes          |
| last_pymnt_amnt     | $7.78 \times 10^0$    | $2.97 \times 10^{-2}$  | yes          |
| total_pymnt         | $6.76 \times 10^0$    | $6.05 \times 10^{-2}$  | yes          |
| total_pymnt_inv     | $6.57 \times 10^0$    | $6.07 \times 10^{-2}$  | yes          |
| int_rate            | $5.14 \times 10^0$    | $3.50 \times 10^{-3}$  | yes          |
| total_rec_int       | $3.07 \times 10^0$    | $5.85 \times 10^{-2}$  | yes          |
| ...                 | ...                   |                        | ...          |
| pymnt_plan          | $2.00 \times 10^{-3}$ | $3.31 \times 10^{-5}$  | no           |
| hardship_flag       | $1.00 \times 10^{-3}$ | $3.31 \times 10^{-5}$  | no           |
| recoveries          | 0                     | $3.31 \times 10^{-5}$  | no           |
| initial_list_status | 0                     | $1.04 \times 10^{-5}$  | no           |
| collection_recover  | 0                     | $3.31 \times 10^{-5}$  | no           |
| policy_code         | 0                     | $1.75 \times 10^{-15}$ | no           |
| acc_now_delinq      | 0                     | $1.75 \times 10^{-15}$ | no           |
| delinq_amnt         | 0                     | $3.12 \times 10^{-6}$  | no           |
| num_tl_30dpd        | 0                     | $1.75 \times 10^{-15}$ | no           |
| pub_rec_bankruptc   | 0                     | $1.82 \times 10^{-5}$  | no           |
| tax_liens           | 0                     | $1.75 \times 10^{-15}$ | no           |

It can be seen from the above Table 5.4 that there are 11 feature variables with IV values less than 0.02. According to the analysis in Section 2.6.3 of Chapter 2, IV values less than

0.02 are of low correlation with the comprehensive performance of the model, so they need to be deleted. At the same time, the mutual information values of these feature variables are also the smallest, indicating that these features have little impact on the final accuracy of the model and can be deleted. The deleted variables are shown in Table 5.5.

**Table 5.5: Feature variables deletion**

| Attribute Name          | IV value              | Whether high |
|-------------------------|-----------------------|--------------|
| pymnt_plan              | $2.00 \times 10^{-3}$ | No           |
| hardship_flag           | $1.00 \times 10^{-3}$ | No           |
| Recoveries              | 0                     | No           |
| initial_list_status     | 0                     | No           |
| collection_recovery_fee | 0                     | No           |
| policy_code             | 0                     | No           |
| acc_now_delinq          | 0                     | No           |
| delinq_amnt             | 0                     | No           |
| num_tl_30dpd            | 0                     | No           |
| pub_rec_bankruptcies    | 0                     | No           |
| tax_liens               | 0                     | No           |

After deleting the variables with low correlation, the remaining 99 variables were put into the model for experiment. The content of the test, cpu usage and other hardware resource control is the same as section 4.2.1. The experimental results are as follow. Figure 5.2 shows the time used by different algorithms after optimization.

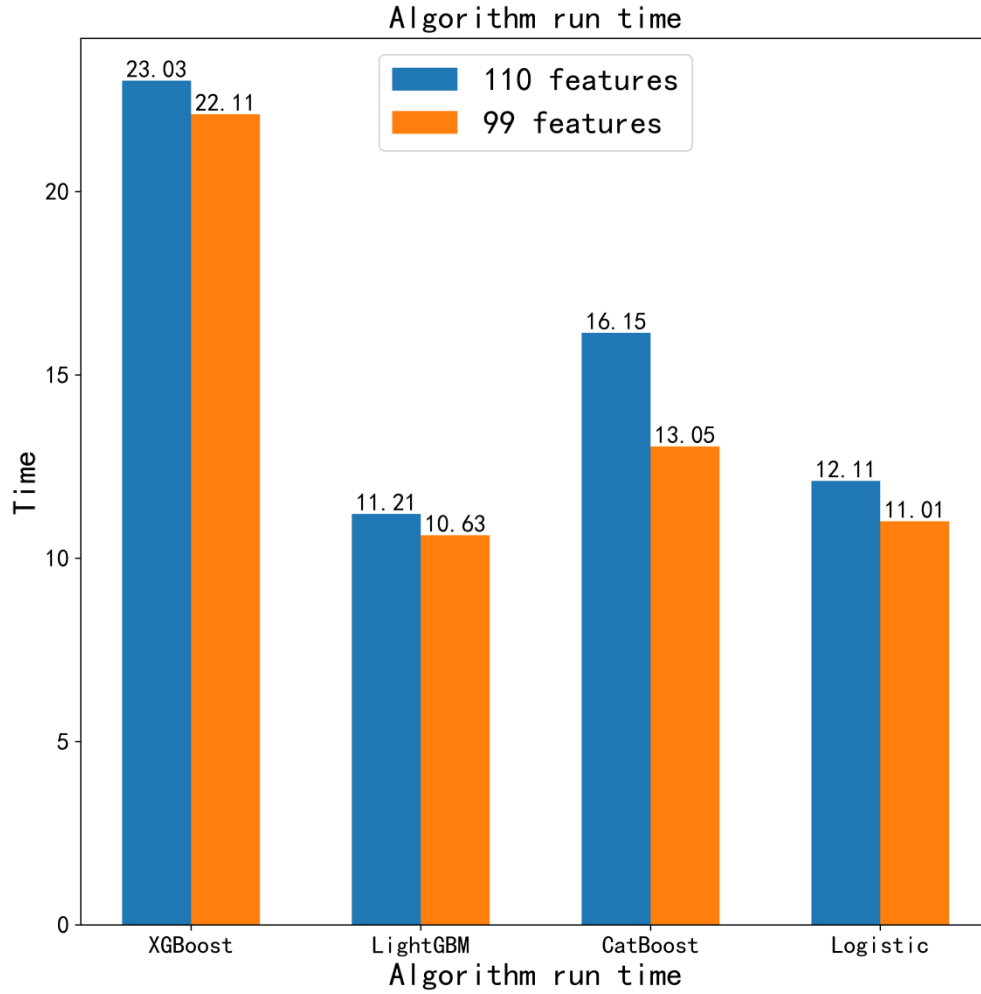


**Figure 5.2: Algorithm running time 99 variables**

In the case of 99 feature variables, the Comprehensive performance of three algorithms (CatBoost, LightGBM and XGBoost) and logistic regression algorithm are shown in Table 5.6. In the table, Recall, AUC, F1 are indicators of the ability to distinguish good and bad customers.

**Table 5.6: Performance indicators of different algorithm (unit: %)**

| Algorithm name | Accuracy | Precision | Recall | AUC   | F1    | Number features |
|----------------|----------|-----------|--------|-------|-------|-----------------|
| CatBoost       | 89.54    | 88.01     | 85.37  | 86.89 | 86.67 | 99              |
| CatBoost       | 89.97    | 88.66     | 86.12  | 87.09 | 87.37 | 110             |

|          |       |       |       |       |       |     |
|----------|-------|-------|-------|-------|-------|-----|
| XGBoost  | 89.31 | 88.03 | 87.49 | 86.37 | 87.76 | 99  |
| XGBoost  | 85.50 | 84.31 | 83.79 | 83.87 | 84.05 | 110 |
| LightGMB | 82.13 | 76.20 | 79.98 | 83.01 | 78.04 | 99  |
| LightGMB | 82.51 | 76.55 | 80.11 | 83.55 | 78.29 | 110 |
| Logistic | 73.07 | 70.88 | 72.00 | 74.34 | 71.44 | 99  |
| Logistic | 73.33 | 71.12 | 72.33 | 74.80 | 71.72 | 110 |

---

#### 5.2.4 Experimental analysis of LendingClub dataset

According to the comparative analysis between Table 5.6, the accuracy and customer differentiation ability of the three algorithms are slightly lower than those without dimension reduction (110 features), but the overall difference is not large, and CatBoost algorithm has the best comprehensive performance. At the same time, it can be seen in Figure 5.2 that the running time of the CatBoost algorithm is not much different from LightGMB algorithm, with a difference of about 2.4 seconds. which is about 20% higher than that before dimension reduction (110 features), and compared with logistic regression algorithm, it is about 2 seconds slower. From Figure 5.2 and Table 5.6, it can be analyzed that, in terms of algorithm time, the XGBoost algorithm takes significantly longer for classification compared to the logistic regression algorithm. Although XGBoost achieves higher accuracy, one crucial feature of a risk control model is that the classification speed of the algorithm should not be slow. Therefore, the XGBoost algorithm does not meet the requirements for classification speed in risk control models.

In Figure 5.2 and Table 5.6, the analysis of the algorithm execution time shows that a key criterion is that the classification speed must be fast enough. The processing time of

XGBoost is significantly longer than that of CatBoost. Therefore, the speed required for effective risk control modeling cannot be met. In terms of accuracy, the accuracy of XGBoost algorithm is lower than that of CatBoost. Therefore, CatBoost algorithm performs better than XGBoost.

Therefore, the test results of the control dataset are consistent with the test results of the China\_Credit dataset, that is, the CatBoost algorithm has high accuracy and good customer differentiation ability, and also has good classification performance (The running time of the algorithm is not much different from that of the logistic regression algorithm).

### **5.3 Conclusion**

In summary, after the comparative test of experiments, CatBoost algorithm is finally selected as the algorithm to replace the traditional risk control model to solve the two problems proposed in Chapter 1, that is, to improve the accuracy of the model and the ability to distinguish customers, and at the same time, it has better classification performance (the algorithm has better running time). It runs about 20% faster than it did before dimensionality reduction (selection variables). Compared with the logistic regression algorithm, the accuracy of the CatBoost algorithm is improved by about 20%, and the classification performance of the two algorithms is similar (the algorithm running time is similar). In addition, in Tables 5.3 and 5.6, the accuracy of the two datasets varies considerably, with the China\_Credit dataset significantly worse than LendingClub because they use the same default parameters. if the CatBoost algorithm wants to further improve the accuracy and distinguish good customers and bad customers (Qi et al., 2021), it needs to adjust its parameters to obtain the optimal parameters. If manually adjusting each parameter of CatBoost, it will be very complexity and time-consuming, so it is necessary to choose an

algorithm that automatically finds the optimal parameters to combine with CatBoost. Solve the problem of algorithmic complexity. This is covered in detail in Chapter 6.

#### **5.4 Chapter Summary**

This chapter focuses on optimizing the classification performance of CatBoost algorithm. By feature selection, the feature variables with low correlation degree are deleted to improve the running speed of CatBoost algorithm and finally improve the classification performance of the algorithm.

## CHAPTER 6

### CATBOOST COMPLEXITY OPTIMIZATION

#### 6.1 Overview

In view of the second question problems raised in section 1.4, that is to further improve the accuracy of the model while reducing the complexity of the risk control model. Corresponding to Section 4.3, the second shortcoming of CatBoost algorithm is proposed, that is, how to further improve the accuracy of CatBoost algorithm, while the complexity of the algorithm is reduced. This chapter introduces and evaluates the Enhanced Binary Grey Wolf Optimizer (EBGWO) and its integration with the CatBoost algorithm, creating the EBGWO-CatBoost hybrid approach.

Sections 6.2 to 6.5 are introduced: EBGWO improves upon the traditional Grey Wolf Optimizer (GWO) by dynamically adjusting the number of leader wolves, enhancing its ability to avoid local optima and balance exploration and exploitation. The results show that EBGWO algorithm has more advantages than GWO algorithm in searching parameters.

Sections 6.6 to 6.7 are introduced: EBGWO algorithm is combined with CatBoost algorithm. The resulting EBGWO-CatBoost algorithm combines the adaptive capabilities of EBGWO with the powerful gradient boosting techniques of CatBoost. Experimental results across various datasets, including China\_Credit, LendingClub, and Portuguese Banking, show that EBGWO-CatBoost significantly outperforms traditional models such as CatBoost, LightGBM, XGBoost, and logistic regression in terms of accuracy and classification performance.

## 6.2 Adaptive Leadership GWO Optimization Algorithm (EBGWO)

According to 5.3, if want to further improve the accuracy of CatBoost algorithm, you need to adjust the parameters of the algorithm. Due to the numerous parameters involved, manually adjusting them one by one is not feasible as it would significantly raise the algorithm's complexity. Therefore, an alternative approach is required to automatically identify the optimal parameters for CatBoost. The Grey Wolf Algorithm (GWO) stands out as a highly effective choice for parameter selection, offering substantial advantages (Gao et al., 2022). Hence, this chapter employs the Grey Wolf Optimizer (GWO) algorithm to optimize the parameters of the CatBoost algorithm, aiming to acquire the most effective set of hyperparameters. By leveraging the GWO algorithm, the objective is to enhance the CatBoost algorithm's performance while simultaneously reducing its time complexity. This optimization process seeks to streamline the model's computational classification performance and improve its predictive accuracy, ultimately leading to more efficient and effective machine learning outcomes.

The Grey Wolf Optimizer (GWO) algorithm is renowned for its simplicity, ease of implementation, and minimal parameter adjustments required. One of its notable advantages lies in its ability to navigate optimization landscapes effectively without getting trapped in local optima. This characteristic makes the GWO algorithm particularly powerful in optimization tasks, allowing it to exhibit prominent optimization capabilities. By virtue of these qualities, the GWO algorithm presents itself as a practical and efficient tool for optimizing complex algorithms such as CatBoost, enhancing their performance while minimizing the risk of convergence to suboptimal solutions (Minh et al., 2023; Chen et al., 2023). However, to address the issue of the GWO algorithm being trapped in local optima

and to enhance its global convergence rate, introduced the principles of chaos theory into GWO, resulting in the creation of CGWO (Dhahavakumar & Gopalan., 2021).

To balance the exploration and exploitation capabilities of the GWO algorithm, this study optimizes the mathematical formulas of the original algorithm and derives a new variant of the GWO algorithm.

In summary, the primary contribution of this study is the development of an enhanced GWO optimization algorithm. This algorithm significantly addresses the inherent limitations of the standard Grey Wolf Optimizer (GWO), improving its ability to avoid local optima and achieving a more effective balance between exploration and exploitation. This advancement offers substantial potential for optimizing algorithms such as CatBoost, thereby contributing to superior performance and classification accuracy across various application domains.

### **6.2.1 EBGWO algorithm**

In Section 2.4, the GWO algorithm was extensively discussed, highlighting that gray wolves rely heavily on the information provided by leading wolves, namely  $\alpha$ ,  $\beta$ , and  $\delta$ , to locate prey. However, fixed number of leading wolves ( $\alpha$ ,  $\beta$ ,  $\delta$ ) being set to 3 can result in a weakened exploration ability for the algorithm and increase susceptibility to falling into local optimal solutions (Hatta et al., 2018). To address this challenge, this study introduces a novel approach for adaptively adjusting the number of leader wolves within the algorithm. This method enables the algorithm to dynamically determine the optimal number of leader wolves based on the prevailing conditions. By doing so, the algorithm can effectively prevent the risk of converging to suboptimal solutions. This enhanced version of the GWO algorithm, incorporating adaptive leader wolf selection, is referred to as the Enhanced Binary Grey Wolf Optimizer (EBGWO) algorithm. The main contribution of EBGWO is to use different

numbers of leaders to update a single position in the iteration process, and to continuously adjust the number of leader wolves through the parameterized adaptive mechanism to strike a balance between exploration and exploitation capabilities. This experimental process is shown in Section 6.2.2.

### **6.2.2 Quantity renewal based on leadership**

The number of leader wolves in the original GWO algorithm is fixed to three, namely  $\alpha$ ,  $\beta$ ,  $\delta$ . As can be seen from Equation 2.5 in Section 2.4.1.3 of Chapter 2,  $X(t+1)$  represents the updated position of the Wolf, which is calculated by  $X(t+1)=(X1+X2+X3)/3$ , and  $X$  is the position vector of the gray Wolf. If the amount of data is very large (the amount of computation is very large), the data samples are irregular, the data segments are somewhat similar and high, and some are scattered and irregular. When the attributes of data samples are easy to distinguish good customers from bad customers, only two leading wolves are needed for calculation ( $X1$  and  $X2$ ), which can improve the convergence speed of the algorithm and reduce the number of iterations. When the data is between good and bad customer samples, the search scope needs to be expanded, and the number of leader wolves needs to be increased, that is, the calculation of  $X1$ ,  $X2$  and  $X3$  is not enough, and an  $X4$  needs to be added. However, the original GWO algorithm performs the calculation of three leading wolves regardless of the data situation, which makes the algorithm inefficient and needs to be further optimized.

According to the problem mentioned above, when the number of leaders increases to four, that is,  $\alpha$  guides  $\beta$ ,  $\delta$  and  $\gamma$  to attack the prey, where  $\alpha$ ,  $\beta$ ,  $\delta$  and  $\gamma$  represent the currently obtained four optimal solutions, and the new individual update formula is as follows:

$$\begin{aligned} D_{\alpha} &= |C_1 \circ X_{\alpha} - X|, \\ D_{\beta} &= |C_2 \circ X_{\beta} - X|, \end{aligned} \quad \text{Equation 6.1}$$

$$\begin{aligned} D_{\delta} &= |C_3 \circ X_{\delta} - X|, \\ D_{\gamma} &= |C_4 \circ X_{\gamma} - X|. \end{aligned}$$

$$X_1 = X_{\alpha} - A_1 \circ D_{\alpha}, \quad \text{Equation 6.2}$$

$$X_2 = X_{\beta} - A_2 \circ D_{\beta},$$

$$X_3 = X_{\delta} - A_3 \circ D_{\delta},$$

$$X_4 = X_{\gamma} - A_4 \circ D_{\gamma}.$$

$$X(t+1) = \frac{X_1 + X_2 + X_3 + X_4}{4} \quad \text{Equation 6.3}$$

Among them,  $D_{\alpha}$ ,  $D_{\beta}$ ,  $D_{\delta}$ ,  $D_{\gamma}$ , represent the distances between the  $\alpha$  wolf,  $\beta$  wolf,  $\delta$  wolf,  $\gamma$  wolf, respectively, and other members of the wolf pack.  $X$  is the current position vector of the gray wolf.  $X_{\alpha}$ ,  $X_{\beta}$ ,  $X_{\delta}$ ,  $X_{\gamma}$  are the positions of the  $\alpha$  wolf,  $\beta$  wolf,  $\delta$  wolf, and  $\gamma$  wolf, respectively.  $A$  and  $C$  are synergy coefficient vectors.  $A$  decreases linearly from 2 to 0 over the entire iterative process.  $r_1$  and  $r_2$  (random vector see 2.4.1.2) are random vectors in the range  $[0,1]$ .  $X(t+1)$  represents the updated position of the wolf after iteration "t+1".

Similarly, reducing the number of leader wolves means that  $\alpha$  wolves guide  $\beta$  wolves to attack prey, and the new individual update formula is:

$$\begin{aligned} D_{\alpha} &= |C_1 \circ X_{\alpha} - X|, \\ D_{\beta} &= |C_2 \circ X_{\beta} - X|, \end{aligned} \quad \text{Equation 6.4}$$

$$X_1 = X_{\alpha} - A_1 \circ D_{\alpha},$$

$$X_2 = X_{\beta} - A_2 \circ D_{\beta}.$$

$$X(t+1) = \frac{X_1 + X_2}{2} \quad \text{Equation 6.6}$$

As previously mentioned, when the number of leader wolves is set to 2, the mathematical formulas of the original GWO algorithm are modified to those in Equation

6.4-6.6. Conversely, when the number of leader wolves is set to 4, the formulas are adjusted to those in Equation 6.1-6.3. The formula for calculating the value of  $X(t+1)$  is not fixed but is automatically selected based on the actual situation for the number of leader wolves. Replacing the original GWO algorithm formula  $X(t+1)=(X1+X2+X3)/3$  with  $X(t+1)=(X1+X2)/2$  or  $X(t+1)=(X1+X2+X3+X4)/4$ . The number of leading wolves for  $X(t+1)$  is either two or four, depending on the value  $|A|$ .  $|A|$  is collaborative coefficient, according to Figure 2.6 of Attacking prey in Section 2.4.1.4 of Chapter 2, when  $|A| \geq 1$  shows the current or A prey's position is not very good, you need to search its prey, It shows that the current calculation amount cannot obtain the optimal solution, that is to say, the three leadership classes of the original GWO algorithm cannot calculate the optimal solution and need to recalculate. It also shows that the differentiation between good and bad customers of the current data sample is very fuzzy, and further calculation is needed to increase, so the number of leader wolves needs to be increased. When the position of the population of prey  $|A| < 1$  is better, need to reduce the number of leader wolves, improve the operation classification performance of the algorithm. When the number of leading wolves changes too much, it will seriously affect the convergence ability of GWO algorithm. Only by reducing or increasing the number of leading wolves according to the actual situation, can we obtain better convergence ability (Heidari et al., 2019), the original GWO algorithm needs to appropriately reduce or increase the number of leader wolves to obtain the optimal result. Therefore, to improve the GWO algorithm. This study adjusts the number of leader wolves based on the size of set A. When  $|A| \geq 1$ , the number of leader wolves increases by 1. So  $X(t+1)$  is the average of the 4 leader wolves. Conversely, when  $|A| < 1$ , the number of leader wolves decreases by 1. So  $X(t+1)$  is the average of the 2 leader wolves. The modified algorithm is detailed in formula (6.7).

$$X(t+1) = \begin{cases} \frac{X_1 + X_2 + X_3 + X_4}{2} & \text{if } |A| > 1 \\ \frac{X_1 + X_2}{2} & \text{else} \end{cases} \quad \text{Equation 6.7}$$

### 6.3 Pseudocode of EBGWO

The pseudocode of the EBGWO algorithm is depicted in Figure 6.1. Unlike the GWO algorithm, where the number of leader wolves ( $\alpha$ ,  $\beta$ ,  $\delta$ ) remains fixed at 3, resulting in limited exploration capabilities, the main contribution of the new EBGWO algorithm proposed in this thesis is the utilization of varying numbers of leaders to update a single position during the iterative process. This enhancement aims to achieve a more balanced trade-off between exploration and exploitation, thereby improving the overall performance of the algorithm.

In steps 10-14 of the pseudo-code algorithm in Figure 6.1, conditional judgment is added and then the corresponding arithmetic expression is executed. The number of leader wolves is constantly adjusted through a adaptive mechanism to solve the problem of weak exploration ability of algorithm.

---

**Algorithm** EBGWO

---

```
1: Initialize the grey wolf population  $X_i (i=1, 2, \dots, n)$ 
2: Initialize  $a$ ,  $A$ , and  $C$ 
3: Calculate the fitness of each search agent
4:  $X_\alpha$  = the best search agent
5:  $X_\beta$  = the best search agent
6:  $X_\delta$  = the best search agent
7:  $X_\gamma$  = the best search agent
8: while ( $t < \text{Max number of iterations}$ )
9:   for each search agent
10:    ▷ Update the position of the current search agent
11:    if ( $|A| < 1$ ):
12:       $X(t+1) = (X_\alpha + X_\beta) / 2$ 
13:    else:
14:       $X(t+1) = (X_\alpha + X_\beta + X_\delta + X_\gamma) / 4$ 
15:  end for
16:  Update  $a$ ,  $A$ , and  $C$ 
17:  Calculate the fitness of all search agents
18:  Update  $X_\alpha$ ,  $X_\beta$ ,  $X_\delta$ , and  $X_\gamma$ 
19:   $t = t + 1$ 
20: end while
21: return  $X_\alpha$ 
```

---

**Figure 6.1: EBGWO algorithm pseudocode**

#### 6.4 Experimental of EBGWO algorithm

The experiment is divided into two parts.

Part one: To clearly explain the impact of number of GWO and EBGWO algorithm leaders on algorithm performance. According to the research of experts Mirjalili et al. (2014), unimodal benchmarking functions are suitable for testing the development ability of algorithms, while multi-modal benchmarking functions are suitable for testing the search ability of algorithms. Therefore, this study uses 20 kinds of benchmarking functions for

algorithm testing, among which F1-F7 is single-peak benchmarking function and F8-F20 is multi-peak benchmarking function.

Part two: The datasets selected for the experiment are the two datasets in Section 3.2.1 which are China\_Credit and LendingClub.

#### 6.4.1 Experimental result of benchmarking test functions and analysis

This study adopts 20 benchmark test functions for simulation experiments. Table 6.1 and Table 6.2 give the basic information of 20 benchmark functions, and the values in the table are referring to Wu et al. (2017), where F1-F7 is an unimodal function and F8-F20 is a multi-modal function. The algorithm parameters are configured as follows: a population size of  $N = 30$ , a maximum number of iterations set to  $T = 1000$ , with termination occurring when the maximum number of iterations is reached. To ensure fair comparisons, each algorithm independently runs for 100 iterations, recording the best value (Best) and standard deviation (Std). These recorded values are then compared with the operational results obtained from both the GWO and EBGWO algorithms. This standardized approach allows for a comprehensive evaluation of the algorithms' performances under consistent conditions. All experiments are implemented in MATLAB-R2012a.

**Table 6.1: Expressions of unimodal functions (F1-F7)**

| Function name | Expression                                          | dimensionality | Range      |
|---------------|-----------------------------------------------------|----------------|------------|
| F1            | $F_1(x) = \sum_{i=1}^D x_i$                         | 30             | [-100,100] |
| F2            | $F_2(x) = \sum_{i=1}^D  x_i  + \prod_{i=1}^D  x_i $ | 30             | [-10,10]   |

|    |                                                                  |    |              |
|----|------------------------------------------------------------------|----|--------------|
| F3 | $F_3(x) = \sum_{i=1}^D (\sum_{j=1}^D x_j)^2$                     | 30 | [-100,100]   |
| F4 | $F_4(x) = \max_{1 \leq i \leq D}  x_i $                          | 30 | [-100,100]   |
| F5 | $F_5(x) = \sum_{i=1}^{D-1} (100(x_{i+1} - x_i^2) - (x_i - 1)^2)$ | 30 | [-30,30]     |
| F6 | $F_6(x) = \sum_{i=1}^D (x_i + 0.5)^2$                            | 30 | [-100,100]   |
| F7 | $F_7(x) = \sum_{i=1}^D ix_i^4 + \text{random}[0,1)$              | 30 | [-1.25,1.25] |

**Table 6.2: Expressions of multimodal functions (F8-F20)**

| Function name | Expression                                                                                                                                                 | Dimensionality | Range        |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|--------------|
| F8            | $F_8(x) = \sum_{i=1}^D x_i \sin(\sqrt{ x_i })$                                                                                                             | 30             | [-100,100]   |
| F9            | $F_9(x) = \sum_{i=1}^D (x_i^2 - 10 \cos(2\pi x_i + 10))$                                                                                                   | 30             | [-5.12,5.12] |
| F10           | $F_{10}(x) = -20 \exp(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^D x_i^2}) - \exp(\frac{1}{D} \sum_{i=1}^D x_i^2 \cos(2\pi x_i)) + 20 + e$                          | 30             | [-32,32]     |
| F11           | $F_{11}(x) = \frac{1}{4000} \sum_{i=1}^D x_i^2 - \prod_{i=1}^D \cos(\frac{x_i}{\sqrt{i}}) + 1$                                                             | 30             | [-600,600]   |
| F12           | $F_{12}(x) = 0.1 \{ \sin^2(\frac{3\pi x_1}{2}) + \sum_{i=1}^{D-1} (100(x_i - 1)^2 [1 + \sin^2(3\pi x_{i+1} + 1)]) + (x_D - 1)^2 [1 + \sin^2(2\pi x_D)] \}$ | 30             | [-50,50]     |
| F13           | $F_{13}(x) = (\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^2 (x_i - a_{ij})^6}) - 1$                                                            | 2              | [-65,65]     |

|     |                                                                                                                                                                        |   |        |
|-----|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---|--------|
| F14 | $F_{14}(x) = \sum_{i=1}^{11} [a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}]^2$                                                                             | 4 | [-5,5] |
| F15 | $F_{15}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$                                                                                          | 2 | [-5,5] |
| F16 | $F_{16}(x) = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos x_1 + 10$                                                           | 2 | [-5,5] |
| F17 | $F_{17}(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2 \times [30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)])]$ | 2 | [-2,2] |
| F18 | $F_{18}(x) = \sum_{i=1}^4 c_i \exp(-\sum_{j=1}^3 a_{ij}(x_i - p_{ij})^2)$                                                                                              | 3 | [1,3]  |
| F19 | $F_{19}(x) = \sum_{i=1}^7 [(X - a_i)(X - a_i)^T + c_i]^{-1}$                                                                                                           | 4 | [0,10] |
| F20 | $F_{20}(x) = \sum_{i=1}^{10} [(X - a_i)(X - a_i)^T + c_i]^{-1}$                                                                                                        | 4 | [0,10] |

WOA (whale optimization algorithm) and GWO algorithm are both intelligent group algorithms. In intelligent group algorithm testing, WOA algorithm has a better comparison effect (Nasiri et al., 2018). In this study, WOA algorithm is compared with GWO and EBGWO algorithm. The comparison process is as follows.

Attacking prey (development stage): This section compares GWO, EB-GWO and WOA algorithms. Table 6.3 shows Best and Std values of three algorithms (GWO, EB-GWO, WOA) in the seven test functions. The Table 6.3 results show that for the best value, EBGWO algorithm this column is better than the original GWO results on 5 test functions (F1-F4 and F6). In the comparison, most of the results obtained from the proposed algorithm outperform those of the Whale Optimization Algorithm (WOA) across all test functions. This indicates that the proposed algorithm is more effective in approaching the theoretical optimal solution. Additionally, it suggests that the proposed algorithm exhibits stronger

development capabilities compared to the WOA. Specifically, when considering the standard deviation (Std) values, the Enhance Balance Grey Wolf Optimizer (EBGWO) algorithm achieves superior results on functions F1, F2, F3, F4, and F7. This further emphasizes the enhanced development abilities of the EBGWO algorithm compared to the standard Grey Wolf Optimizer (GWO). In terms of Std values, EB-GWO gets better results on F1, F2, F3, F4, and F7 functions.

Research for prey (Exploration stage): Functions F8-F20 are multimodal functions. They include many local optimizations, and the number of local optimizations increases exponentially as the dimension increases. Therefore, these functions are considered to be very useful in evaluating the exploration capabilities of algorithms (Mirjalili et al., 2014). Table 6.4 lists the Best and Std values of the three comparison algorithms on F8-F20. As can be seen from the table, the Best Value of EB-GWO is the smallest on 10 functions (F9, F11, and F13-F20, Bold font in the table), the Best Value of GWO is the smallest on 8 functions (F9, F11, F13-F18), and the Best Value of WOA is also the smallest on 10 functions (F8-F13, F15-F18). For Std values, EB-GWO has the smallest Std value on 7 functions (F8-F10, F13, F17-F18, F20), GWO has the smallest Std value on 3 functions (F11, F14, F19), and WOA has the smallest Std value on 4 functions (F9, F12, F15-F16). From the analysis of test results, EBGWO is the best in the test results of most test functions, and the proposed algorithm has strong competitiveness in solving multi-peak problems. This also proves that EB-GWO has good exploration capability.

**Table 6.3: Test results on unimodal functions (F1-F7)**

| Function name | GWO  | EB-GWO | WOA  | GWO | EB-GWO | WOA |
|---------------|------|--------|------|-----|--------|-----|
|               | Best | Best   | Best | std | std    | Std |

|    |                                         |                                           |                       |                                         |                                           |                                         |
|----|-----------------------------------------|-------------------------------------------|-----------------------|-----------------------------------------|-------------------------------------------|-----------------------------------------|
| F1 | $1.14 \times 10^{-60}$                  | <b><math>-8.70 \times 10^{-74}</math></b> | 0                     | $1.32 \times 10^{-59}$                  | <b><math>-1.05 \times 10^{-71}</math></b> | 0                                       |
| F2 | $2.14 \times 10^{-35}$                  | <b><math>-1.19 \times 10^{-43}</math></b> | 0                     | $5.93 \times 10^{-35}$                  | <b><math>-9.00 \times 10^{-43}</math></b> | 0                                       |
| F3 | $1.42 \times 10^{-19}$                  | <b><math>1.44 \times 10^{-23}</math></b>  | $4.95 \times 10^{10}$ | $9.91 \times 10^{-15}$                  | <b><math>1.65 \times 10^{-15}</math></b>  | $7.34 \times 10^3$                      |
| F4 | $9.22 \times 10^{-16}$                  | <b><math>7.27 \times 10^{-19}</math></b>  | $6.24 \times 10^{-3}$ | $1.20 \times 10^{-15}$                  | <b><math>2.94 \times 10^{-17}</math></b>  | $1.60 \times 10^1$                      |
| F5 | <b><math>2.61 \times 10^1</math></b>    | $2.62 \times 10^1$                        | $2.66 \times 10^1$    | $7.03 \times 10^{-1}$                   | $6.33 \times 10^{-1}$                     | <b><math>3.65 \times 10^{-1}</math></b> |
| F6 | $2.49 \times 10^{-1}$                   | <b><math>8.20 \times 10^{-5}</math></b>   | $8.10 \times 10^{-3}$ | <b><math>2.95 \times 10^{-1}</math></b> | $4.17 \times 10^{-1}$                     | $7.95 \times 10^{-2}$                   |
| F7 | <b><math>1.80 \times 10^{-4}</math></b> | $2.4 \times 10^{-4}$                      | $7.00 \times 10^{-5}$ | $5.00 \times 10^{-4}$                   | <b><math>3.94 \times 10^{-4}</math></b>   | $2.14 \times 10^{-3}$                   |

**Table 6.4: Test results on multimodal functions (F8-F20)**

| Function name | GWO                                     | EB-GWO                                  | WOA                                      | GWO                                     | EB-GWO                                  | WOA                                     |
|---------------|-----------------------------------------|-----------------------------------------|------------------------------------------|-----------------------------------------|-----------------------------------------|-----------------------------------------|
|               | Best                                    | Best                                    | Best                                     | std                                     | std                                     | std                                     |
| F8            | $-6.72 \times 10^{-3}$                  | $-6.73 \times 10^{-3}$                  | <b><math>-7.70 \times 10^{-3}</math></b> | $1.14 \times 10^3$                      | <b><math>6.31 \times 10^2</math></b>    | $7.22 \times 10^2$                      |
| F9            | <b>0</b>                                | <b>0</b>                                | <b>0</b>                                 | $1.88 \times 10^0$                      | <b>0</b>                                | <b>0</b>                                |
| F10           | $1.50 \times 10^{-14}$                  | $7.99 \times 10^{-15}$                  | <b><math>8.88 \times 10^{-16}</math></b> | $2.39 \times 10^{-15}$                  | <b>0</b>                                | $2.80 \times 10^{-15}$                  |
| F11           | <b>0</b>                                | <b>0</b>                                | <b>0</b>                                 | <b><math>4.27 \times 10^{-3}</math></b> | $8.86 \times 10^{-3}$                   | $1.61 \times 10^{-2}$                   |
| F12           | $9.489 \times 10^{-2}$                  | $2.13 \times 10^{-1}$                   | <b><math>8.24 \times 10^{-2}</math></b>  | $2.43 \times 10^{-1}$                   | $1.48 \times 10^{-1}$                   | <b><math>8.08 \times 10^{-2}</math></b> |
| F13           | <b><math>9.98 \times 10^{-1}</math></b> | <b><math>9.98 \times 10^{-1}</math></b> | <b><math>9.98 \times 10^{-1}</math></b>  | $9.12 \times 10^{-1}$                   | <b><math>8.36 \times 10^{-1}</math></b> | $9.58 \times 10^{-1}$                   |
| F14           | <b><math>3.10 \times 10^{-4}</math></b> | <b><math>3.10 \times 10^{-4}</math></b> | $3.20 \times 10^{-4}$                    | <b><math>1.13 \times 10^{-7}</math></b> | $2.88 \times 10^{-5}$                   | $4.91 \times 10^{-4}$                   |
| F15           | <b><math>-1.03 \times 10^0</math></b>   | <b><math>-1.03 \times 10^0</math></b>   | <b><math>-1.03 \times 10^0</math></b>    | $3.83 \times 10^{-9}$                   | $4.29 \times 10^{-9}$                   | <b><math>1.36 \times 10^{-9}</math></b> |
| F16           | <b><math>3.97 \times 10^{-1}</math></b> | <b><math>3.97 \times 10^{-1}</math></b> | <b><math>3.97 \times 10^{-1}</math></b>  | $5.83 \times 10^{-9}$                   | $2.00 \times 10^{-6}$                   | <b><math>2.48 \times 10^{-7}</math></b> |
| F17           | <b><math>3.00 \times 10^0</math></b>    | <b><math>3.00 \times 10^0</math></b>    | <b><math>3.00 \times 10^0</math></b>     | $1.00 \times 10^{-5}$                   | <b><math>4.00 \times 10^{-6}</math></b> | $5.00 \times 10^{-6}$                   |
| F18           | <b><math>-3.86 \times 10^0</math></b>   | <b><math>-3.86 \times 10^0</math></b>   | <b><math>-3.86 \times 10^0</math></b>    | $5.30 \times 10^{-5}$                   | <b><math>6.00 \times 10^{-6}</math></b> | $4.97 \times 10^{-4}$                   |
| F19           | $-1.04 \times 10^1$                     | <b><math>-1.05 \times 10^1</math></b>   | $-1.04 \times 10^{-1}$                   | <b><math>2.25 \times 10^{-4}</math></b> | $3.00 \times 10^{-4}$                   | $2.81 \times 10^0$                      |
| F20           | $-1.05 \times 10^1$                     | <b><math>-1.06 \times 10^1</math></b>   | $-1.05 \times 10^1$                      | $4.11 \times 10^{-4}$                   | <b><math>3.33 \times 10^{-4}</math></b> | $3.23 \times 10^0$                      |

According to the research of experts (Ribeiro et al., 2020; Ahmed et al., 2023), the two benchmark functions of Shpere and Schwefel can effectively test the performance of global search and local search of intelligent optimization algorithms. Therefore, the following two

benchmark functions will be selected to visually show the GWO and EBGWO development phase and exploration phase.

The parameter Settings of the two functions are shown in Table 6.5. In the table, the population size of N (dimension), the upper boundary of D, the lower boundary of i, and X are vectors. The parameter values in the table are obtained by referring to the research of experts Yao et al. (1999). Since the original default number of leaders of the GWO algorithm is 3, and the EBGWO algorithm is mainly for the adjustment and optimization of the number of leaders, the experiment variables are mainly conducted according to the number of different leaders.

**Table 6.5: Parameter settings for shpere and schwefel functions**

| Function name | Expression                                   | Range      | D  | N  | T <sub>max</sub> |
|---------------|----------------------------------------------|------------|----|----|------------------|
| Shpere        | $F_1(x) = \sum_{i=1}^D x$                    | [-100,100] | 30 | 20 | 100              |
| Schwefel      | $F_8(x) = \sum_{i=1}^D x \sin(\sqrt{ x_i })$ | [-100,100] | 30 | 20 | 100              |

#### 6.4.2 Experimental results and analysis of Shpere and Schwefel functions

To determine the appropriate number of iterations, experimental tests are conducted across a range of iterations, from 0 to 100, using the Sphere and Schwefel benchmark functions. The results of these experimental tests are depicted in Figure 6.2, providing insights into the algorithms' performance at different iteration counts. Based on the observed trends and convergence behavior exhibited in the figures, the number of iterations can be

selected accordingly. This iterative process allows for the refinement of the optimization algorithm's parameters to achieve optimal performance on the given benchmark functions.

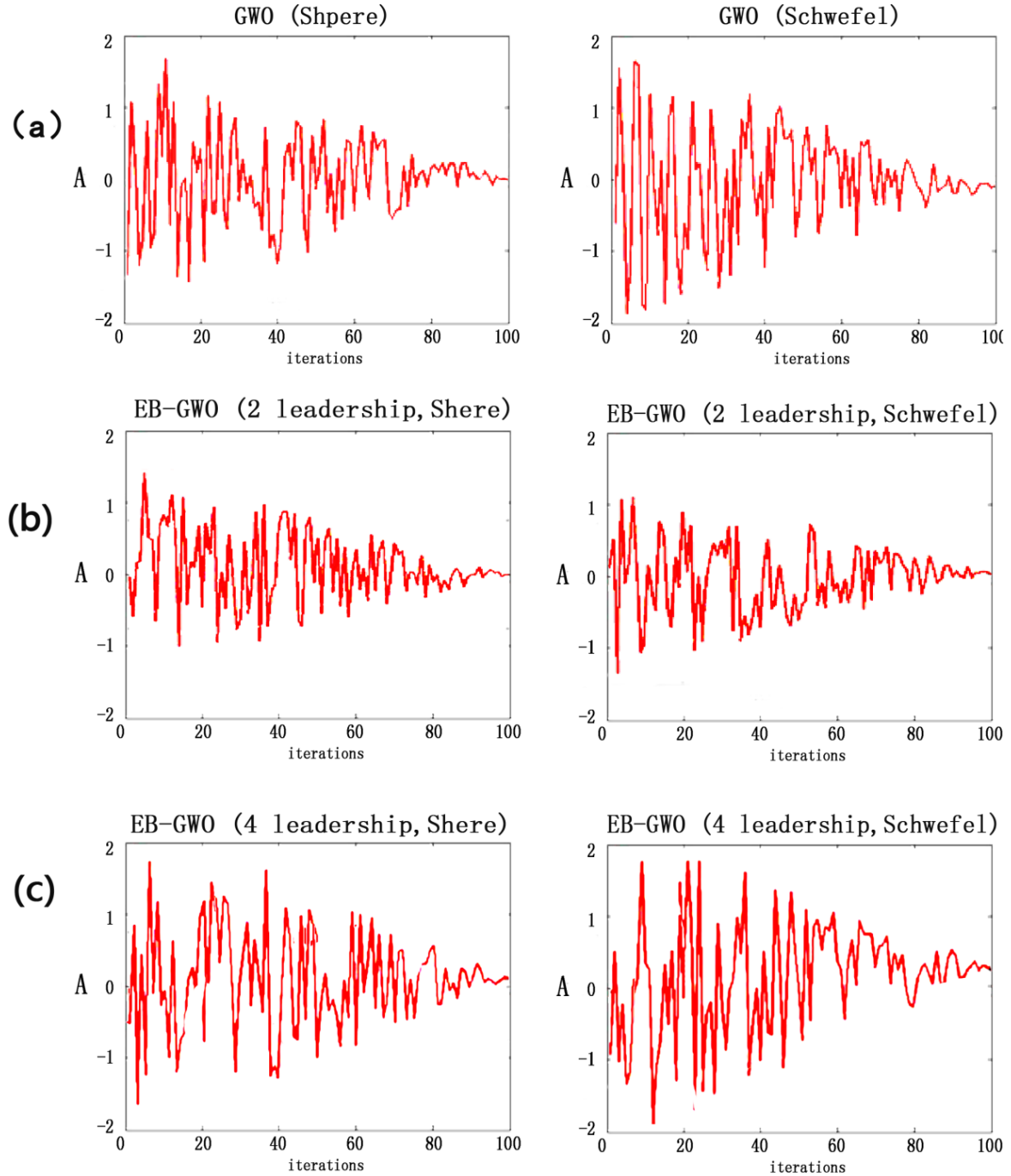


**Figure 6.2: Tests of A in shpere and schwefel functions**

It can be seen from Figure 6.2 above that when the number of iterations reaches about 100, the A value tends to be stable, so it is reasonable to adopt 100 iterations in this study.

From the GWO algorithm analysis in section 2.4.2.2, the value of  $A$  directly affects the update position of the leadership wolves, so the stability of the value of  $A$  means the stability of the leadership wolves.

From Figure 6.2 of the search history of the two functions that the solutions in GWO and EBGWO gradually approach the direction of the optimal solution in the evolution process, and eventually converge to the global optimal.

The value of  $A$  directly affects the position update of leader wolves, so the larger the range of  $A$ , the larger the range of wolves' position movement and update, that is, the larger the search scope. Row (a) in Figure 6.2 shows the experimental results of the GWO algorithm of three leaders wolves. Row (b) is the experimental results of EBGWO algorithm of 2 leaders wolves; Row (c) is the experimental results of EBGWO algorithm of 4 leaders wolves; Compared row (b) and (c), in terms of the search history of gray wolves in 100 iterations on Shpere and Schwefel functions, the EBGWO with the number of leaders wolves at 4 has a wider (large amplitude) and more diverse track distribution than that with the number of leaders wolves at 2. This shows that the algorithm has a strong exploration ability when the number of leaders is large. According to section 2.4.2.4 attacking prey (development stage),  $|A| < 1$ , it indicates that wolves are in the attacking prey and have strong development capability. The fluctuation amplitude of  $A$  in row (b) is smaller than that in row (c), and most of the fluctuations are less than 1, so the development stage is relatively strong when the number of leaders wolves is 2. The experimental results of row (a) (GWO) are exactly between row (b) and row (c) (EBGWO), so the GWO algorithm is weaker than EBGWO algorithm in both development and exploration capability.

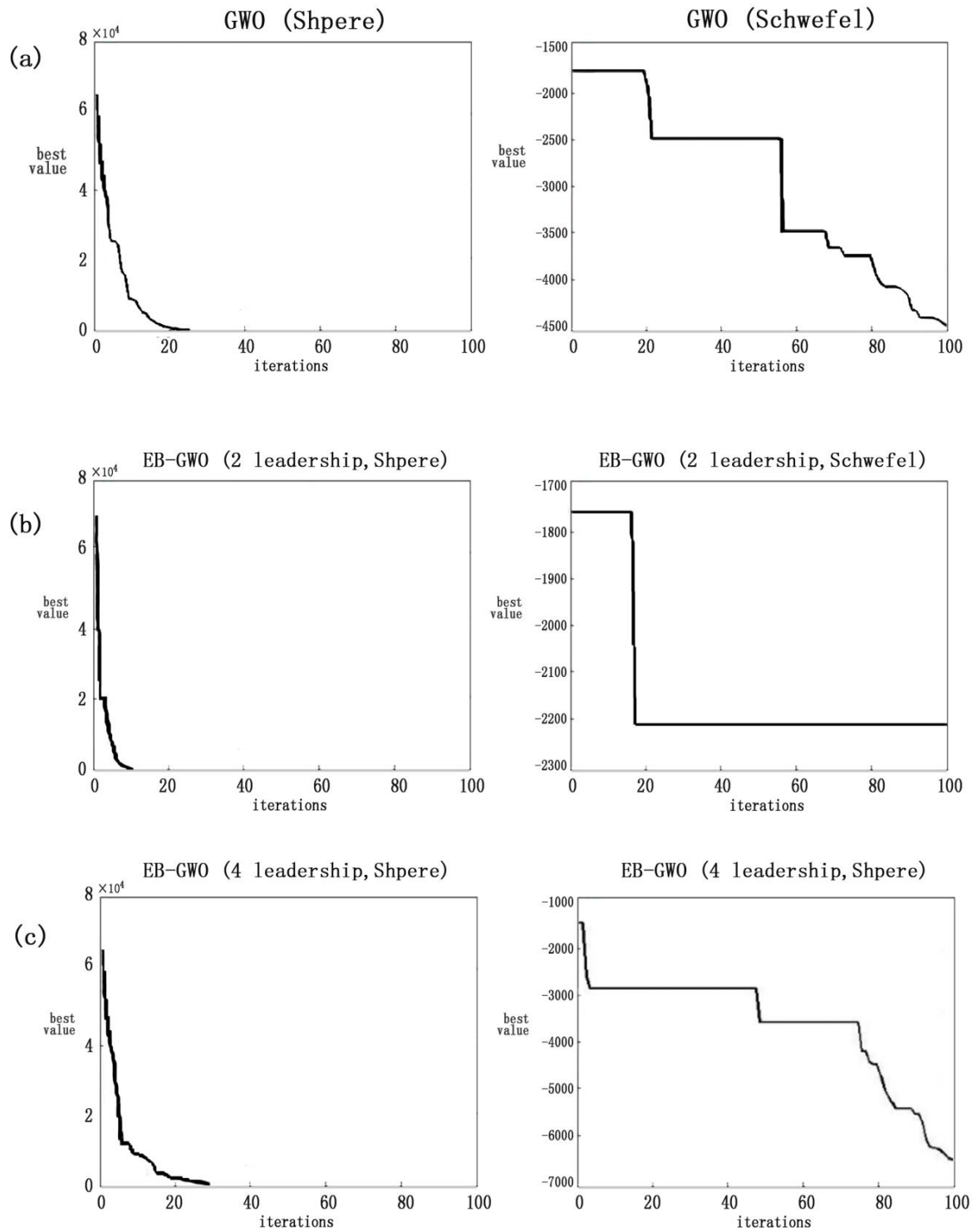


**Figure 6.3: Tests of best value on Shere and Schwefel functions**

By comparing the experimental best value results of the GWO algorithm (number of leaders wolves is 3) and EBGWO algorithm (number of leaders wolves is 2 or 4), row (a) in

Figure 6.3 that the best value calculated when the number of iterations ranges from 0 to 20 and 20 to 60 is a straight line, which is easy to fall into the local optimal value. Of course, when the number of iterations reaches more than 70 times, the global optimal value is gradually obtained. Compared with Figure 6.3 row (b) in and row (c) of the EBGWO algorithm, it can be seen from the optimal value that when the number of leaders wolves is 2, the global optimal solution converges 20 iterations, and the convergence time is very fast. When leader wolves number is 4 (Figure 6.3 row (c)), it is similar to leaders wolves number is 3 (Figure 6.3row (a)), its tracks were longer and larger, so they had greater exploration capabilities. Compared with Figure 6.3 row (a) and row (c), EBGWO is exploration capabilities are stronger than the GWO algorithm. Increasing the number of iterations beyond 100 can help reduce the risk of falling into local optima by allowing the algorithm to explore the search space more thoroughly and find better solutions. However, doing so may significantly increase the computational time required to achieve convergence. Therefore, the number of iterations must be selected judiciously based on the specific requirements of the optimization task and the available computational resources.

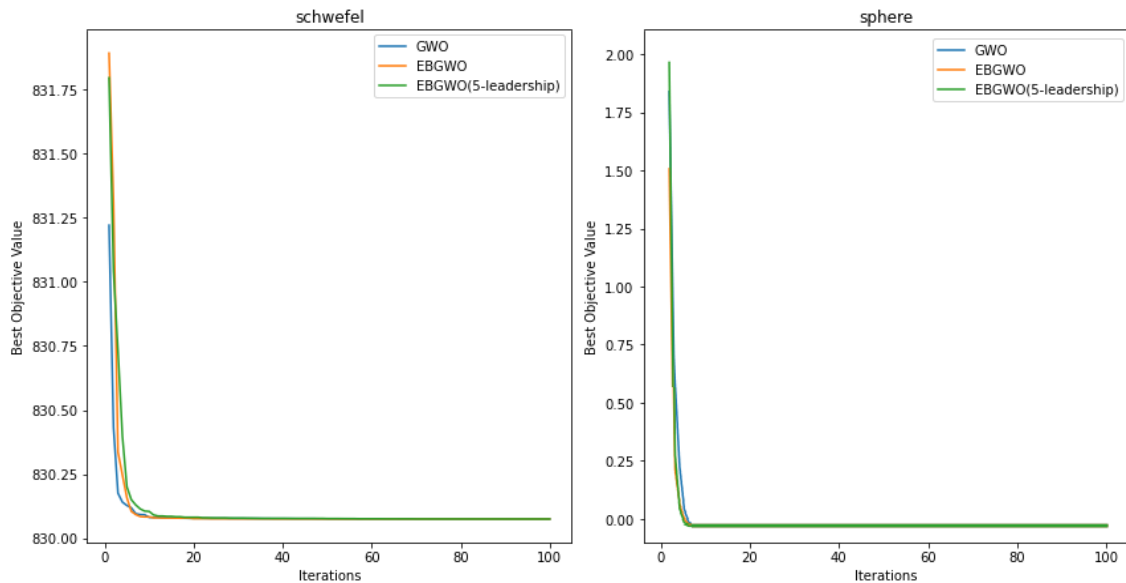


**Figure 6.4: Leadership wolves number is 5 test**

From the analysis of Figure 6.4, it is observed that "EBGWO(5-leadership)" represents the GWO algorithm with 5 leader wolves, shown in green, while the GWO curve is in blue and the EBGWO algorithm is in orange. It can be seen that the orange line (EBGWO) almost overlaps with the EBGWO(5-leadership) line, indicating that increasing the number of leader wolves from 4 to 5 has nearly the same effect, and more leaders actually increase the algorithm's runtime ("EBGWO (5-leadership)" line in the figure has the highest number of iterations). Therefore, it can be concluded that the GWO algorithm does not benefit from having more leader wolves; rather, adding one extra leader wolf achieves the best results.

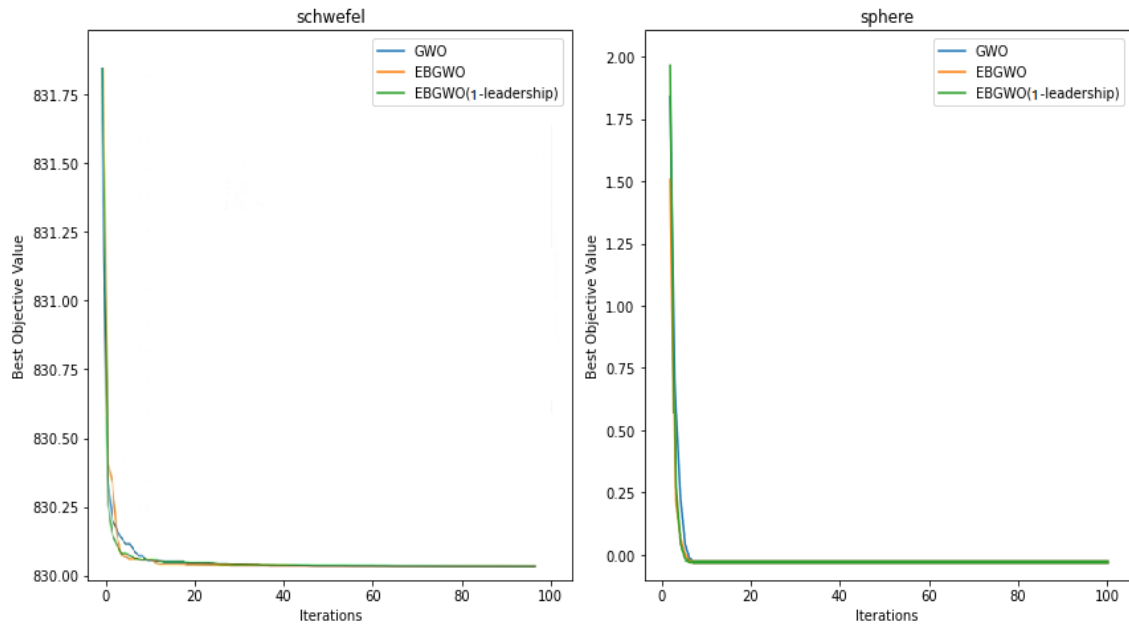


**Figure 6.5: Leadership wolf number is 1 test**

Figure 6.5 shows the experimental results when the number of leader wolves is 1. The graph indicates that two lines are above the green line (with 1 leader wolf) until the iteration count reaches about 10, at which point they converge into a single line. The green line has close to 5 iterations, indicating it is the fastest. According to the study by Heidari et al. (2019),

a lower number of leader wolves in the GWO algorithm leads to lower accuracy. Therefore, reducing the number of leaders to 2 is the most appropriate for the GWO algorithm.

### 6.4.3 Experimental results of China\_Credit dataset

In this section, the comprehensive performance of the Enhance Balance Grey Wolf Optimizer (EBGWO) algorithm will be verified using the China\_Credit dataset. Combine the GWO and EBGWO algorithms with the logistic regression algorithm, naming them “GWO-LR” and “EBGWO-LR,” and then conduct experiments. Both the GWO-LR and EBGWO-LR algorithms will be applied to train models on this dataset, and their performance will be evaluated. The evaluation method described in Section 3.2.3 of Chapter 3 will be utilized for this purpose.

The China\_Credit dataset, as described in Section 3.2.1, will be used for training and testing the models. Additionally, the running time of the algorithms will be recorded and presented in Table 6.6 to provide insights into their computational classification performance.

**Table 6.6: Algorithm running time**

| algorithm | time  |
|-----------|-------|
| GWO-LR    | 6.76s |
| EBGWO-LR  | 4.33s |

It can be seen from the experiments in section 6.4.2 that the optimal values of GWO and EBGWO algorithms tend to be stable around 100 iterations. Therefore, in the test of the China\_Credit dataset, 100 iterations are taken as the baseline, and then 20, 50, 80, 100, 120, 150, 180 iterations are carried out respectively, show Table 6.7.

**Table 6.7: Model validation test results**

| name      | 20<br>iteration   | 50<br>iteration   | 80<br>iteration   | 100<br>base line  | 120<br>iteration  | 150<br>iteration  | 180<br>iteration  |
|-----------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|
|           | Accuracy<br>/ AUC | Accuracy /<br>AUC | Accuracy<br>/ AUC | Accuracy<br>/ AUC | Accuracy<br>/ AUC | Accuracy<br>/ AUC | Accuracy<br>/ AUC |
| GWO       | 69.5/69.0         | 70.7/69.8         | 71.8/70.8         | 72.5/71.3         | 72.8/72.0         | 72.7/72.2         | 72.9/72.1         |
| EB<br>GWO | 70.5/69.9         | 71.9/71.3         | 72.4/71.2         | 72.4/71.2         | 72.5/72.0         | 72.4/71.8         | 72.4/71.9         |

#### 6.4.4 Experimental analysis of China\_Credit dataset

Table 6.6, the test time of the improved EBGWO algorithm in the China\_Credit dataset was less than that of the GWO-LR algorithm, which also confirms that the EBGWO-LR algorithm can adjust the number of leader wolves according to the actual situation, thus improving the classification performance of the algorithm.

Table 6.7, with 100 iterations as the baseline, GWO-LR tends to be stable (optimal solution) only after 100 iterations, while the EBGWO-LR algorithm tends to be stable about 80iterations, which also indicates that the EBGWO algorithm is more efficient in finding parameters than the GWO algorithm.

However, as can also be analyzed from Table 6.7, neither GWO nor EBGWO algorithm combined with logistic regression algorithm can improve the accuracy of traditional risk control model.

#### 6.4.5 Experimental results of LendingClub dataset

In this section, this study will verify the comprehensive performance of the Enhance Balance Grey Wolf Optimizer (EBGWO) algorithm using the LendingClub dataset. As in Section 6.4.3, both the GWO-LR and EBGWO-LR algorithms will be used to train models

on this dataset, and their performance will be evaluated. The evaluation method described in Section 3.2.3 of Chapter 3 will be applied for this purpose.

The LendingClub dataset, as described in Section 3.2.1, will be used for training and testing the models. Additionally, the running time of the algorithms will be recorded and presented in Table 6.8 to provide insights into their computational classification performance.

**Table 6.8: Algorithm running time**

| algorithm | Time   |
|-----------|--------|
| GWO-LR    | 10.85s |
| EBGWO-LR  | 8.91s  |

This section, 100 iterations are chosen as the baseline. Therefore, in the test of the LendingClub dataset, 100 iterations are taken as the baseline, and then 20, 50, 80, 100, 120, 150, 180 iterations are carried out respectively, show Table 6.9.

**Table 6.9: Model validation test results**

| name      | 20<br>iteration   | 50<br>iteration   | 80<br>iteration   | 100<br>base line  | 120<br>iteration  | 150<br>iteration  | 180<br>iteration  |
|-----------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|
|           | Accuracy<br>/ AUC | Accuracy /<br>AUC | Accuracy<br>/ AUC | Accuracy<br>/ AUC | Accuracy<br>/ AUC | Accuracy<br>/ AUC | Accuracy<br>/ AUC |
| GWO       | 71.3/70.4         | 71.9/71.0         | 72.5/71.9         | 73.5/72.7         | 73.7/72.8         | 73.6/72.9         | 73.7/72.7         |
| EB<br>GWO | 72.2/71.8         | 74.7/74.4         | 75.7/74.4         | 75.8/74.4         | 75.6/74.5         | 75.7/74.4         | 75.7/74.4         |

#### 6.4.6 Experimental analysis of LendingClub dataset

The results in Tables 6.8 and 6.9 are the same as the conclusions in Section 6.4.4, the comparison reveals that EBGWO algorithm outperforms the GWO algorithm in parameter

optimization classification performance. However, despite its classification performance, the EBGWO algorithm exhibits lower accuracy levels, consistent with the analysis in Section 6.4.4. In pursuit of the objective outlined in Section 1.5 of Chapter 1, which aims to enhance the accuracy of the risk control model, the EBGWO-LR algorithm falls short as it fails to meet the required accuracy standards. Therefore, EBGWO-LR cannot be considered a viable option for improving the accuracy of the risk control model. Alternative approaches or algorithms with better accuracy performance should be explored to achieve the desired improvement in model accuracy effectively.

## **6.5 Conclusion of EBGWO algorithm**

Based on the experimental analysis conducted above, it can be concluded that the Grey Wolf Optimizer (GWO) algorithm, with its fixed number of leader wolves set at 3, is prone to falling into local optima. This limitation arises because the GWO algorithm lacks the capability to adaptively adjust the number of leader wolves based on the current situation. The test analysis performed on 20 benchmarking functions demonstrates that the fixed number of leader wolves (leading wolves is 3), it impedes the algorithm's ability to effectively explore and exploit the search space. Consequently, the GWO algorithm struggles to timely adjust its exploration and exploitation balance, leading to suboptimal solutions and a higher likelihood of getting trapped in local optima. The number of leaders wolves in the modified EBGWO algorithm is not fixed but selected according to the actual number of leaders wolves, which makes the algorithm converge quickly and avoids embedding the algorithm into the local optimal solution. At the same time, this feature is also shown in the test of the risk loan dataset. When the data type is between good and bad, the algorithm chooses to leaders the number of wolves to be 2. At this time, the development ability of the algorithm is strong (attack prey stage), which can make the algorithm converge

quickly. The data between good and bad needs to increase the number of leading wolves (4 leading wolves), so that the algorithm is further in the search stage so that the leaders wolves can choose more accurate positions, so as to improve the probability of successful hunting of wolves, that is, you can better distinguish these data between good and bad. It can be seen from the test results of the risk loan dataset that the accuracy of the algorithm is not high, and it cannot distinguish good customers from bad customers very well, so it cannot replace the algorithm of the traditional risk control model, but its parameter searching ability is very strong (Long et al., 2019).

If EBGWO algorithm is combined with CatBoost algorithm, the complexity problem of CatBoost algorithm can be solved, that is, it is impossible to find parameters automatically, but it needs to find parameters manually, and the algorithm complexity is high. This is covered in detail in the next section.

## **6.6 EBGWO-CatBoost algorithm**

GWO algorithm has strong convergence performance and good ability in searching parameters (Arora et al., 2019). From section 6.5, EBGWO Compared with GWO algorithm, EBGWO has better convergence ability, and can not easy fall into local optimal solution, and effectively balances the development and exploration ability of the algorithm (see section 6.4). Therefore, this chapter will try to combine the EBGWO algorithm and the CatBoost algorithm together, called the "EBGWO-CatBoost" algorithm. The good automatic parameter training ability of EBGWO is used to automatically find the optimal parameters of the CatBoost algorithm, which can not only solve the complex problems of the algorithm, but also further improve the accuracy of the model.

Section 6.6.1 of this chapter describes the EBGWO-Catboost combination algorithm code in detail, and Section 6.6.2 describes experimental content, the use of "China\_Credit" dataset for the experiment, and the evaluation of the performance of EBGWO-Catboost algorithm. then, the "LendingClub" dataset is used to test and compare, and the comprehensive performance of the EBGWO-Catboost combination algorithm is optimal.

### 6.6.1 Pseudocode of EBGWO-CatBoost

The core pseudo-code of EBGWO-CatBoost algorithm is shown in Figure 6.6.

---

#### **Algorithm**    EBGWO-Catboot

---

```

1: # Initialize all parameters of catboost(catboost_params) and training data
2: def CatBoost_score(**catboost_params):
3:     # Define global variables and record optimal parameters
4:     global Params_best={ All CatBoost_parameters }
5:     # Initialize all parameters of CatBoost
6:     CatBoost = CatBoostRegressor(**catboost_params)
7:     CatBoost.fit(x_train, y_train)
8:     compare = CatBoost.predict(x_test)
9:     if compare > F_score:
10:         F_score=compare
11:         Params_bes = catboost_params
12:     return 1 - accuracy_score(y_test, predict)
13: # EBGWO-CatBoost Algorithm
14: def EBGWO-CatBoost(CatBoost_score, lb, ub, dim, SearchAgents_no, Max_iter):
15:     # initialize Alpha_pos, Beta_pos, Delta_pos and r_pos
16:     Alpha_pos, Beta_pos, Delta_pos, r_pos = np.zeros(dim)
17:     Alpha_score, Beta_score, Delta_score, r_score = float("inf")
18:     # Initialize all Wolf positions
19:     Positions = np.zeros((SearchAgents_no, dim))
20:     for i in range(dim):
21:         Positions[:, i] = np.ones(SearchAgents_no)*ub[i]
22:     Convergence_curve = np.zeros(Max_iter)
23:     # Iterate to find the optimal value
24:     for l in range(0, Max_iter):
25:         for i in range(0, SearchAgents_no):

```

```

26:         # Return search agents that exceed the boundaries of the search space
27:         for j in range(dim):
28:             Positions[i, j] = np.clip(Positions[i, j], lb[j], ub[j])
29:         # Compute the CatBoost_score function for each search agent
30:         fitness = CatBoost_score(Positions[i, :])
31:         if fitness < Alpha_score:
32:             Alpha_score = fitness
33:             Alpha_pos = Positions[i, :].copy()
34:         if (fitness > Alpha_score and fitness < Beta_score):
35:             Beta_score = fitness
36:             Beta_pos = Positions[i, :].copy()
37:         if (fitness > Alpha_score and fitness > Beta_score and fitness < Delta_score):
38:             Delta_score = fitness
39:             Delta_pos = Positions[i, :].copy()
40:         if (fitness > Alpha_score and fitness > Beta_score and fitness > Delta_score and fitness < r_score):
41:             r_score = fitness
42:             r_pos = Positions[i, :].copy()
43:     a = 2 - 1 * ((2) / Max_iter); # a decreases linearly from 2 to 0
44:     for i in range(0, SearchAgents_no):
45:         for j in range(0, dim):
46:             # X1,X2,X3,X4:  $\alpha, \beta, \delta, \gamma$ , derived next generation grey Wolf position vector
47:             r1,r2 = random.random() # r1,r2 is a random number in [0,1]
48:             A1,A2,A3,A4 = 2 * a * r1 - a
49:             C1,C2,C3,C4 = 2 * r2
50:             D_alpha,D_beta,D_delta,D_r = abs(Ci * Di_pos[j]-Positions[i, j])
51:             X1 = Alpha_pos[j] - A1 * D_alpha , X2 = Beta_pos[j] - A2 * D_alpha
52:             X3 = Delta_pos[j] - A3 * D_alpha , X4 = r_pos[j] - A4 * D_alpha
53:             if (|A|<1):
54:                 Positions[i, j] = (X1 + X2) / 2
55:             else:
56:                 Positions[i, j] = (X1 + X2 + X3 + X4) / 4
57:     Convergence_curve[1] = Alpha_score

```

---

**Figure 6.6: EBGWO-CatBoost algorithm pseudocode**

The EBGWO-CatBoost combination algorithm consists of two parts, the first part is to create the CatBoost\_score function in lines 1-12. The second part is the combination of EBGWO algorithm and CatBoost algorithm, that is, the establishment of "EBGWO\_CatBoost" function in lines 9-57.

Lines 1-12 represent the creation of the CatBoost\_score function, which initializes all parameters of the CatBoost algorithm, training data, and records the best parameters. The variable name "catboost\_params" in line 4 is the best parameter to record after each iteration of CatBoost. The first parameter in the EBGWO-CatBoost function in line 14, "CatBoost\_score", is used to call the CatBoost\_score function.

Lines 9-57 represent the combination of the EBGWO algorithm and the CatBoost algorithm, and the code function name is "EBGWO-CatBoost". Where steps 15-17 represent the values of the initial alpha\_pos, beta\_pos, delta\_pos, r\_pos (that is,  $\alpha$ ,  $\beta$ ,  $\delta$ ,  $\gamma$ ), where float("inf") represents "plus or minus infinity", and all numbers are less than +inf; Positive infinity: float("inf"); Negative infinity: float("-inf"). Lines 18-22 indicate where all wolves are initialized. Lines 23-57 indicate the search for the optimal value, where lines 27-42 indicate the return of a search agent beyond the boundaries of the search space. Lines 43-56 represent the hunting process of the pack, where "Alpha, Beta, Delta, r" in line 50 represents the distance between the candidate and the " $\alpha$ ,  $\beta$ ,  $\delta$ ,  $\gamma$ " Wolf. And the "abs()" function returns the absolute value of the number. Alpha\_pos [j] indicates the Alpha position and Positions [i,j]) positions of the candidate Gray Wolf. In lines 43-45, X1, X2, X3, and X4 are the next generation Grey Wolf position vectors based on Alpha, Beta, Delta, and r, respectively. Lines 46-50 write the core code of the EBGWO algorithm optimization process in Chapter 4, as detailed in Figure 6.1 in Section 6.3.

To calculate the total number of times “CatBoost\_score” is called during the optimization process in the “EBGWO” function. Its formula is 6.8.

$$\text{Total Calls} = \text{Max\_iter} \times \text{SearchAgents\_on} \quad \text{Equation 6.8}$$

Here’s a step-by-step explanation: 1. Number of Iterations (“Max\_iter”): This is the total number of iterations the Grey Wolf Optimizer will perform. 2. Number of Search Agents (“SearchAgents\_no”): This is the number of search agents (wolves) used in each iteration. Since “CatBoost\_score” is called once per agent per iteration, the total number of calls is the product of these two values.

For example Calculation: If “Max\_iter” is 100 and “SearchAgents\_no” is 30, then: Total Calls =  $100 \times 30 = 3000$ . Thus, the function “CatBoost\_score” will be called 3000 times throughout the entire optimization process.

### 6.6.2 Experimental of EBGWO-CatBoost algorithm

The experiment of EBGWO-CatBoost algorithm is similar to Chapter 5, the two datasets used China\_Credit and LendingClub, and used feature variables after dimension reduction (deleting variables with low correlation). For details, see section 5.2.1 and 5.2.3. The EBGWO algorithm will automatically find the best parameters of the Catboost algorithm.

### 6.6.3 Experimental results of China\_Credit dataset

The model trained by CatBoost algorithm and EBGWO-CatBoost algorithm was tested, and then compared with CatBoost, LightGBM, XGBoost and logistic regression algorithms, and finally the model was evaluated. In the experiment, the operating memory, CPU usage and other hardware condition resources of the five algorithms are set as in section 4.2. 1. The experimental results are shown in Figure 6.7.

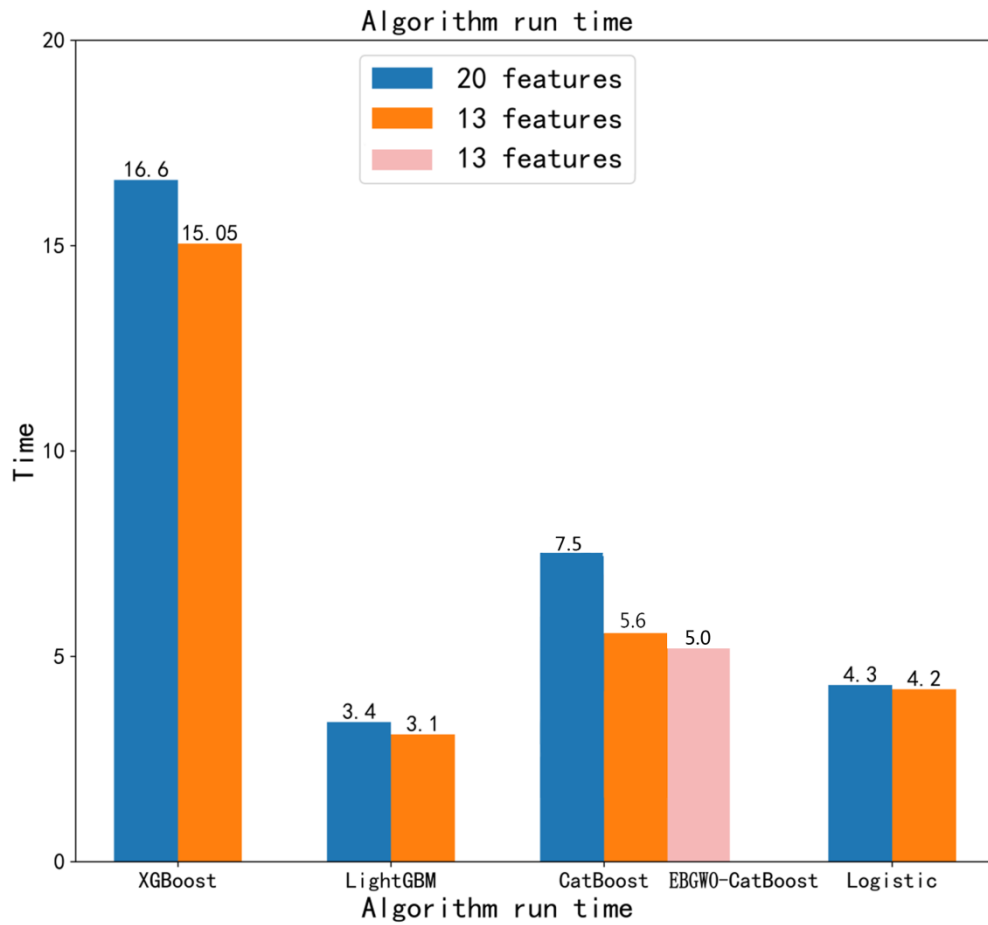


**Figure 6.7: Algorithm run time (13 variables)**

To validate the accuracy of the EBGWO-CatBoost model, comparison experiments are conducted with four other algorithms (CatBoost, Logistic Regression, LightGBM, and XGBoost). The accuracy of these tests is summarized in Table 6.10.

**Table 6.10: Model performance evaluation Table (unit: %)**

| Algorithm name | Accuracy     | Precision    | Recal        | AUC          | F1           | Number features |
|----------------|--------------|--------------|--------------|--------------|--------------|-----------------|
| EBGWO-CatBoost | <b>84.05</b> | <b>84.00</b> | <b>85.04</b> | <b>85.02</b> | <b>41.51</b> | <b>13</b>       |
| EBGWO-CatBoost | 84.07        | 83.92        | 85.11        | 85.24        | 41.41        | 20              |
| CatBoost       | 82.05        | 82.71        | 82.51        | 82.03        | 41.24        | 13              |

|          |       |       |       |       |       |    |
|----------|-------|-------|-------|-------|-------|----|
| CatBoost | 82.01 | 81.06 | 82.42 | 82.01 | 40.42 | 20 |
| XGBoost  | 81.44 | 79.92 | 80.02 | 79.54 | 40.79 | 13 |
| XGBoost  | 81.92 | 80.27 | 80.31 | 79.84 | 41.05 | 20 |
| EBGWO-   | 82.20 | 81.13 | 81.65 | 81.34 | 40.91 | 20 |
| LightGMB | 73.06 | 74.27 | 72.64 | 73.57 | 37.11 | 13 |
| LightGMB | 73.36 | 73.00 | 72.81 | 73.02 | 36.72 | 20 |
| EBGWO-   | 74.39 | 74.10 | 73.55 | 74.38 | 37.26 | 20 |

---

Here, the two algorithms "EBGWO-CatBoost" and "CatBoost" with higher accuracy are selected to compare the ROC curves, as shown in Figure 6.8.

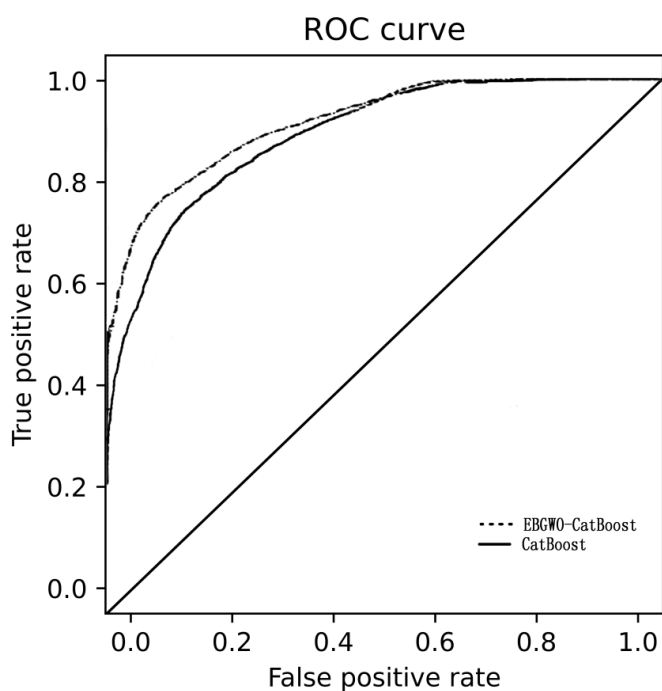


**Figure 6.8: Model ROC Curve**

Use the "EBGWO" algorithm for multiple iterations to find the best CatBoost parameters. Train the CatBoost classifier to get the optimal parameters. Finally, the main parameters affecting the results are determined as follows Table 6.11.

**Table 6.11: EBGWO-CatBoost main parameters**

| parameter                      | Description                                                                                                          |
|--------------------------------|----------------------------------------------------------------------------------------------------------------------|
| iterations=600                 | Maximum number of decision trees.                                                                                    |
| learning_rate=0.02             | The smaller the training time, the higher the number of iterations.                                                  |
| depth=7                        | Depth of the tree.                                                                                                   |
| random_seed=2019               | The number of random seeds during training.                                                                          |
| subsample=0.7                  | Logloss or Cross Entropy is used for binary classification problems, and MultiClass is used for multiclassification. |
| loss_function='Logloss'        | The proportion of training samples.                                                                                  |
| eval_metric=accuracy           | Parameters to prevent overfitting.                                                                                   |
| one_hot_max_size=3             | Unique heat coding.                                                                                                  |
| bagging_temperature=1          | Bayesian bagging control strength.                                                                                   |
| rsm=1                          | Random subspace method.                                                                                              |
| lef_estimation_method=Gradient | Method of calculating leaf value, Newton/Gradient.                                                                   |
| fold_len_multiplier=3          | Folds length coefficient.                                                                                            |

From Table 6.11, the parameters are explained as follows: 1. iterations=600. Description: Sets the maximum number of boosting iterations (trees) to be used. Details: More iterations can potentially lead to better model performance, but also increase the risk of overfitting. It's often tuned based on cross-validation performance. 2.learning\_rate=0.02. Description: Controls the step size of each tree's contribution to the final model. Details: A smaller learning rate means that each tree has a smaller effect, which often requires more iterations to converge but can improve the model's generalization ability. Conversely, a larger learning rate might speed up training but can lead to overfitting. 3. depth=7. Description: Defines the maximum depth of the decision trees in the ensemble. Details:

Deeper trees can model more complex relationships in the data but may lead to overfitting if not managed carefully. Shallower trees are less prone to overfitting but might underfit if too shallow.

4. `random_seed=2019`. Description: Sets the random seed for reproducibility. Details: Using a fixed random seed ensures that the results are consistent across different runs with the same data and parameters. This helps in debugging and comparing models.

5. `subsample=0.7`. Description: Controls the fraction of samples used to fit each tree. Details: A value less than 1 introduces randomness by using only a portion of the training data for each tree, which helps prevent overfitting. For example, a subsample of 0.7 means 70% of the training data is used for each tree.

6. `loss_function='Logloss'`. Description: Specifies the loss function used for model optimization. Details: For binary classification, 'Logloss' (Logarithmic Loss) is appropriate, as it measures how well the predicted probabilities match the actual class labels. For multiclass classification, you might use 'MultiClass'.

7. `eval_metric=accuracy`. Description: Defines the metric used to evaluate the model's performance on the validation set. Details: Accuracy measures the proportion of correctly classified instances out of the total instances. For different problems, you might choose other metrics like AUC, precision, recall, or F1-score.

8. `one_hot_max_size=3`. Description: Limits the number of categories for one-hot encoding. Details: If a categorical feature has fewer unique values than this threshold, CatBoost will apply one-hot encoding. For features with more unique values, CatBoost uses alternative encoding methods to handle high cardinality efficiently.

9. `bagging_temperature=1`. Description: Controls the strength of Bayesian bagging (or stochastic gradient boosting). Details: A higher temperature introduces more randomness in the training data sampling for each tree, which can help in reducing overfitting. A value of 1 is the default and balanced setting; values greater than 1 increase randomness.

10. `rsm=1`. Description: Specifies the fraction of features to be randomly

selected for each tree. Details: When `rsm` is set to 1, all features are used for training each tree. Lower values mean that only a subset of features is considered for each tree, which can help with overfitting and improve model performance on large datasets. 11. `leaf_estimation_method='Gradient'`. Description: Determines the method used to estimate the value of each leaf in the trees. Details: The 'Gradient' method uses gradient descent to optimize leaf values, which is standard. 'Newton' can be used for more accurate second-order optimization but is more computationally intensive. 12. `fold_len_multiplier=3`. Description: Affects the length of the folds in cross-validation. Details: This parameter adjusts the size of each fold relative to the total number of samples. A multiplier of 3 means each fold is 3 times the size of the average fold, which can be useful for ensuring that each fold has a sufficient number of samples for training and validation.

#### **6.6.4 Experimental discuss of China\_Credit dataset**

Result analysis: From the time complexity analysis, it can be seen from Figure 6.5 that the EBGWO-CatBoost combination algorithm takes little time difference from the CatBoost algorithm. Under the same number of iterations, the EBGWO-CatBoost algorithm (13 features) takes 0.6 seconds longer than the CatBoost algorithm (13 features), and only 1 second longer than the traditional logistic regression algorithm (13 features). The running time of the EBGWO-CatBoost algorithm is not much different from that of the traditional logistic regression algorithm, so the new algorithm can replace the old algorithm (logistic regression) in running time.

From Figure 6.5 and Table 6.10, it can be seen that after removing features with low relevance using IV, the algorithm's execution time (classification speed) is somewhat reduced, while the algorithm's accuracy remains almost unchanged. This observation indicates that IV can enhance the classification performance of the algorithm. According to

the study by experts (Raymaekers et al., 2021), the main roles of WOE are: 1. improving model stability, 2. enhancing model robustness, 3. optimizing model performance, and 4. handling missing values. The WOE method used in this paper mainly plays a role in the following aspects: 1. Improving Model Stability: By using binning, continuous variables can be converted into categorical variables. This conversion helps to improve model stability. Even as time progresses and there are slight changes in the target population (e.g., income increasing or decreasing, aging, etc.), the model's scores should remain relatively stable, thus avoiding significant fluctuations in the model. 2. Enhancing Model Robustness: Binning increases the model's tolerance for outliers. For instance, if a sample's monthly income is significantly below the median income of the overall sample, binning can categorize this outlier into a reasonable range. This categorization reduces the impact of outliers on the model. 3. Optimizing Model Performance: Through appropriate binning, the model can better capture the underlying patterns in the data. This improvement enhances the model's predictive accuracy. WOE values reflect the proportion of good and bad samples within each bin. This reflection helps the model to better distinguish between good and bad customers, thereby enhancing the model's discriminatory power.

In Table 6.10, among all algorithms, no matter EBGWO-Catboost algorithm uses 20 features or 13 features, its accuracy is the highest, above 84%. That is, the number of good or bad customers correctly predicted accounted for more than 84% of the total sample. CatBoost and XGBoost ranked second and third, respectively, with an improvement of more than 15% over the traditional model Logistic regression algorithm. The EBGWO-Catboost algorithm improves by about 4% compared to the Catboost algorithm. The recall rate was about 85%; When the accuracy rate is similar to the recall rate, the F1 value is used to judge, and the F1 value of the EBGWO-Catboost algorithm is about 85%, which is also the highest

among the four algorithms. The larger AUC, the closer it is to 1, the better the prediction effect. As shown in Figure 6.6, compared with other algorithms, the AUC value of the EBGWO-Catboost algorithm is the highest, which indicates that its prediction effect is the best.

The analysis in Table 6.10 shows that the accuracy of the CatBoost algorithm is significantly higher than that of the traditional logistic regression algorithm, and the accuracy of the EBGWO-CatBoost algorithm is also higher than that of the CatBoost algorithm. Other differentiation capabilities such as Recall, AUC, and F1 indicators, also show that the EBGWO-CatBoost algorithm is better than the CatBoost algorithm. From Table 6.10, it can be seen that the accuracy of the EBGWO-XGBoost algorithm has improved, but the accuracy of the EBGWO-XGBoost algorithm still falls short compared to the accuracy of the EBGWO-CatBoost algorithm. Although the accuracy of the EBGWO-LightGBM algorithm has also improved, the accuracy of the EBGWO-LightGBM algorithm is still far behind the accuracy of the EBGWO-CatBoost algorithm. Therefore, the combination of EBGWO and the CatBoost algorithm has better comprehensive performance than a single algorithm. Therefore, the algorithm using the EBGWO-CatBoost combination algorithm to optimize the traditional risk control model has the best comprehensive performance and can replace the traditional logistic regression algorithm.

### **6.6.5 Experimental results of LendingClub dataset**

For the stability and generality of the new algorithm, this section uses another dataset to test and compare, and the experiment is performed in the same way as the step in 6.6.2. With the same number of iterations, the experimental time of EBGWO-CatBoost, LightGB, XGBoost, CatBoost and Logistic regression is shown in Figure 6.9.

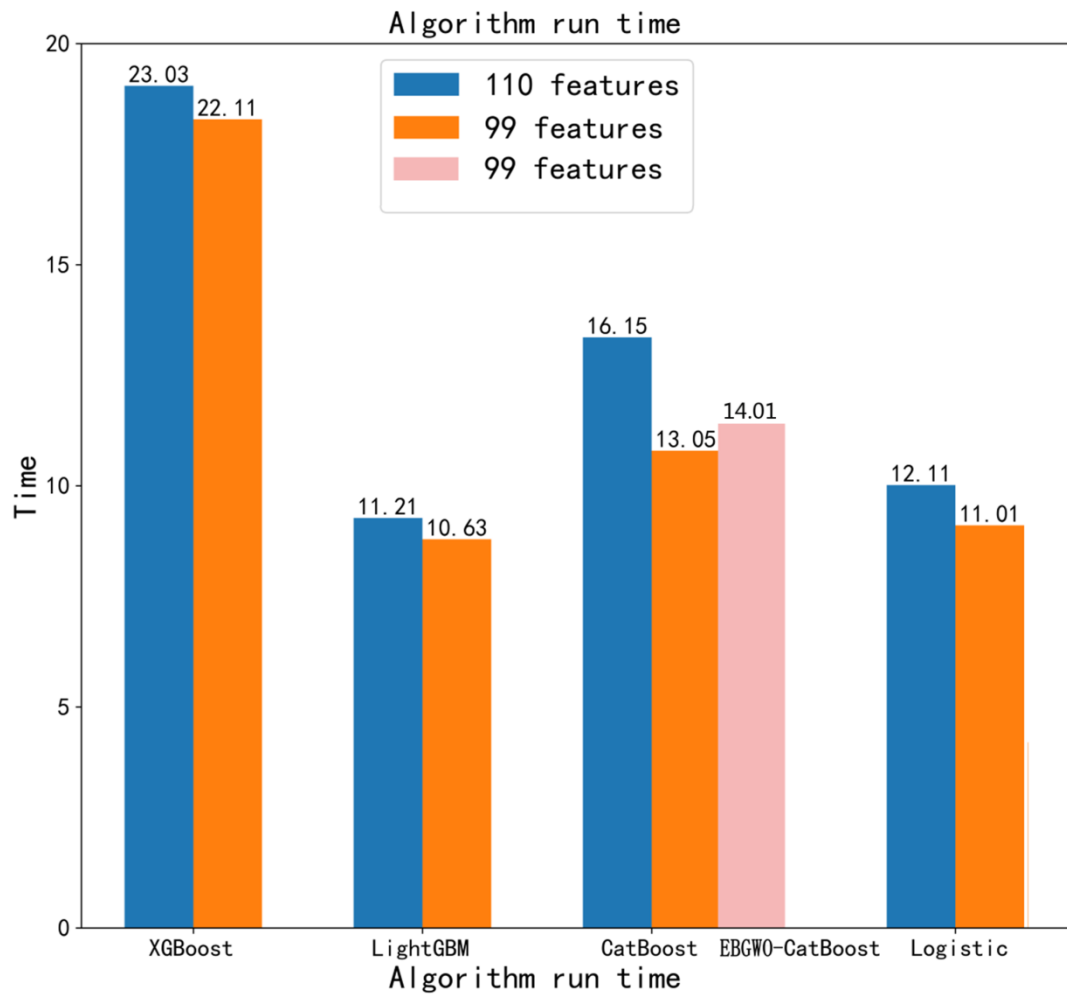


**Figure 6.9: Time profile of different algorithms running**

In order to verify the accuracy of EBGWO-CatBoost model, CatBoost, Logistic, LightGBM and XGboost algorithms are respectively used to conduct experimental comparison tests with EBGWO-CatBoost. The accuracy of the tests is shown in Table 6.12.

**Table 6.12: Model performance evaluation (unit: %)**

| Algorithm name | Accuracy     | Precision    | Recall       | AUC          | F1           | Number features |
|----------------|--------------|--------------|--------------|--------------|--------------|-----------------|
| EBGWO-CatBoost | <b>92.44</b> | <b>92.00</b> | <b>90.53</b> | <b>91.05</b> | <b>46.83</b> | <b>99</b>       |
| EBGWO-CatBoost | 92.53        | 92.02        | 90.74        | 91.88        | 46.62        | 110             |
| CatBoost       | 89.54        | 88.01        | 85.37        | 86.89        | 45.74        | 99              |

|          |       |       |       |       |       |     |
|----------|-------|-------|-------|-------|-------|-----|
| CatBoost | 90.32 | 89.71 | 86.94 | 86.11 | 46.82 | 110 |
| EBGWO-   | 89.97 | 88.66 | 86.12 | 87.09 | 46.05 | 110 |
| XGBoost  | 89.31 | 88.03 | 87.49 | 86.37 | 45.22 | 99  |
| XGBoost  | 85.50 | 84.31 | 83.79 | 83.87 | 42.99 | 110 |
| EBGWO-   | 82.99 | 81.99 | 81.55 | 80.86 | 41.89 | 110 |
| LightGMB | 82.13 | 81.31 | 79.98 | 80.01 | 41.74 | 99  |
| LightGMB | 82.71 | 81.45 | 81.11 | 80.87 | 41.59 | 110 |

Here, the two algorithms "EBGWO-CatBoost", "CatBoost", "XGBoost" with higher accuracy are selected to compare the ROC curves, as shown in Figure 6.10.

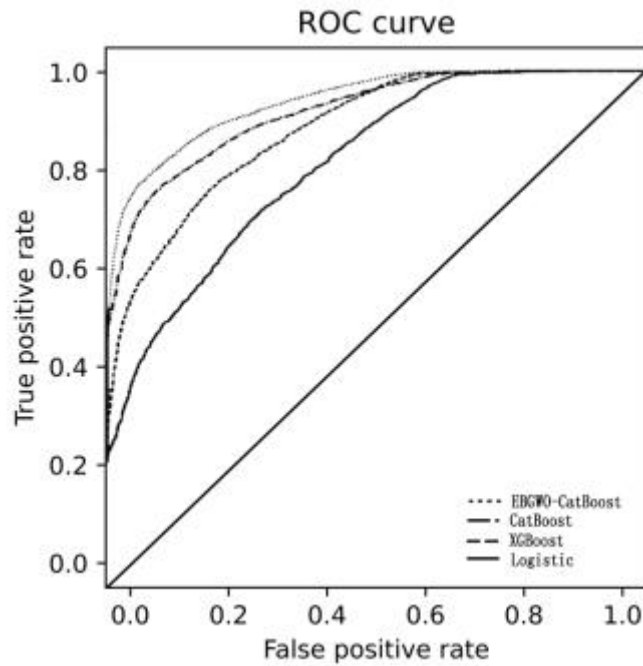


**Figure 6.10: Model ROC Curve**

Combining Figures 6.7 and 6.8, it can analyze that the EBGWO-CatBoost algorithm has the largest area under the ROC curve in Figure 6.8, which indicates that this algorithm's performance is the best. Additionally, analyzing the algorithm's runtime in Figure 6.7 shows

that, compared to traditional logistic regression, the EBGWO-CatBoost algorithm is only about 3 seconds slower. Therefore, it achieves a good balance of high accuracy and relatively fast runtime (classification performance). This balance meets the current risk control model's requirements for high accuracy and quick algorithm execution. On the other hand, the XGBoost algorithm also has a high accuracy but takes too long, being twice as slow as the original logistic regression algorithm. Thus, it does not meet the requirement for fast algorithm speed in risk control models. Finally, while the LightGBM algorithm is the fastest and has higher accuracy than logistic regression. But LightGBM's accuracy is nowhere near as high as CatBoost's. From the perspective of traditional risk control model requirements, which prioritize higher accuracy, this algorithm is also excluded. In summary, the EBGWO-CatBoost algorithm satisfies the current risk control model requirements in terms of both algorithm runtime (classification performance) and accuracy.

Use the "EBGWO" algorithm for multiple iterations to find the best CatBoost parameters. Train the CatBoost classifier to get the optimal parameters. Finally, the main parameters affecting the results are determined as follows Table 6.13. The details of the parameters in the table are similar to the explanations provided in Table 6.11 of Section 6.6.3.

**Table 6.13: EBGWO-CatBoost main parameters**

|                    |                                                      |
|--------------------|------------------------------------------------------|
| Training data      | Test data                                            |
| iterations=200     | Maximum number of decision trees.                    |
| learning_rate=0.03 | The smaller the training time, the higher the number |
| depth=6            | Depth of the tree.                                   |
| random_seed = 2018 | The number of random seeds during training.          |
| subsample=0.6      | Logloss or Cross Entropy is used for binary          |

|                              |                                                    |
|------------------------------|----------------------------------------------------|
| loss_function= 'Logloss'     | The proportion of training samples.                |
| eval_metric=accuracy         | Parameters to prevent overfitting.                 |
| one_hot_max_size=2           | Unique heat coding.                                |
| bagging_temperature=1        | Bayesian bagging control strength.                 |
| rsm=1                        | Random subspace method.                            |
| lef_estimation_method=Gradie | Method of calculating leaf value, Newton/Gradient. |
| fold_len_multiplier=4        | Folds length coefficient.                          |

### 6.6.6 Experimental discuss of LendingClub dataset

Result analysis: From the analysis of time complexity, it can be seen from Figure 6.7 that the EBGWO-CatBoost combination algorithm takes little time difference from the CatBoost algorithm. Under the same number of iterations, the EBGWO-CatBoost algorithm takes 1 second longer than the CatBoost algorithm (99 features), and only 3 seconds longer than the traditional logistic regression algorithm (99 features). The new algorithm is almost the same as the traditional logistic regression algorithm, so the new algorithm can replace the old algorithm in running time.

From Figure 6.7 and Table 6.12, it can be seen that by removing features with low correlation using IV (Information Value), there is a certain reduction in algorithm time (classification speed), while the accuracy of the algorithm remains nearly unchanged. This indicates that IV can improve the classification performance of the algorithm.

From the analysis of Table 6.12 and Figure 6.7, among all algorithms, no matter EBGWO-Catboost algorithm uses 110 features or 99 features, its accuracy is the highest, above 90%. That is, the number of good or bad customers correctly predicted accounted for more than 90% of the total sample. With an improvement of more than 25% over the

traditional model Logistic regression algorithm. The EBGWO-Catboost algorithm improves by about 4% compared to the Catboost algorithm. it is also proved that the EBGWO-CatBoost combination algorithm has the best comprehensive performance and can replace the algorithm of the traditional risk control model. In Table 6.10, it can also be observed that the EBGWO-XGBoost algorithm shows an improvement in accuracy, but it still does not match the accuracy of the EBGWO-CatBoost algorithm. Although the EBGWO-LightGBM algorithm also shows improved accuracy, it is still far behind the EBGWO-CatBoost algorithm. Through the verification and comparison of the LendingClub dataset, it is consistent with the results of the China\_Credit dataset test.

#### **6.6.7 Experimental results and discuss of other dataset**

To test the generalizability of the algorithm, this study selected credit-related datasets from the UCI Machine Learning Repository.

The data is direct marketing campaigns (phone calls) of a Portuguese banking. The basic information of the dataset is: 2008-2010 years has a total of more than 40000 data and 20 variables<sup>3</sup>. The data is related with direct marketing campaigns (phone calls) of a Portuguese banking institution. The classification goal is to predict if the client will subscribe a term deposit (variable y). The data is related with direct marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls. Often, more than one contact to the same client was required, in order to access if the product (bank term deposit) would be ('yes') or not ('no') subscribed.

---

<sup>3</sup><http://archive.ics.uci.edu/static/public/222/bank+marketing.zip>

The features of the dataset include age, job, destiny, education, default, balance, housing, loan, contact, day\_of\_week, month, duration, campaign, updates, previous, upcoming, and 16 additional features. In the dataset, ‘y’ represents the results (good or bad customers). Dimensionality reduction was performed using the method described in Section 2.6.3. Subsequently, three variables with low correlation—namely age, marital status, and contact—were removed. This thesis employs the experimental methods outlined in Sections 6.6.3 and 6.6.5 to test the dataset. The results of these tests are presented in Table 6.14.

**Table 6.14: Model performance evaluation (unit: %)**

| Algorithm name | Accuracy     | Precision    | Recal        | AUC          | F1           | Number features |
|----------------|--------------|--------------|--------------|--------------|--------------|-----------------|
| EBGWO-CatBoost | <b>83.86</b> | <b>82.53</b> | <b>83.99</b> | <b>83.15</b> | <b>41.41</b> | <b>13</b>       |
| EBGWO-CatBoost | 84.01        | 83.49        | 84.71        | 84.23        | 41.52        | 16              |
| CatBoost       | 82.07        | 81.13        | 82.05        | 81.41        | 40.73        | 13              |
| CatBoost       | 82.01        | 81.06        | 82.42        | 82.01        | 40.42        | 16              |
| XGBoost        | 81.20        | 80.34        | 81.01        | 79.88        | 40.54        | 13              |
| XGBoost        | 82.19        | 81.17        | 81.12        | 80.43        | 41.29        | 16              |
| EBGWO-XGBoost  | 82.36        | 81.32        | 81.75        | 81.09        | 41.13        | 16              |
| LightGMB       | 77.76        | 76.81        | 77.15        | 77.99        | 38.49        | 13              |
| LightGMB       | 78.14        | 77.27        | 77.64        | 78.37        | 38.70        | 16              |
| EBGWO-LightGMB | 79.02        | 78.75        | 78.97        | 77.88        | 39.67        | 16              |
| Logistic       | 72.71        | 72.02        | 72.85        | 72.51        | 36.02        | 13              |
| Logistic       | 73.03        | 72.61        | 73.02        | 72.83        | 36.56        | 16              |

The test results show that the EBGWO-CatBoost algorithm has the highest accuracy and performs best across other metrics as well. Its accuracy is approximately 3% higher than that of the CatBoost algorithm, 2% higher than both the EBGWO-XGBoost and XGBoost algorithms, and about 10% higher than the EBGWO-LightGBM and Logistic algorithms. Therefore, these results are consistent with those in Sections 6.6.4 and 6.6.6.

## **6.7 Conclusion of EBGWO-CatBoost algorithm**

Through the experiment and analysis of the two datasets. From the accuracy analysis, regardless of whether the EBGWO-CatBoost algorithm carries out dimensionality reduction (feature screening), its accuracy is increased by about 4% compared with the CatBoost algorithm. Compared with the logistic regression algorithm, its accuracy is increased by more than 15%. From the analysis of algorithm classification performance, it can be seen that if dimension reduction (feature screening) is used, the EBGWO-CatBoost algorithm can improve its accuracy by about 20% compared with logistic regression algorithm operation without affecting its own accuracy. The comprehensive performance of EBGWO-CatBoost algorithm is the best compared with CatBoost, logistic regression, LightGMB and Xgboost algorithms.

The experimental results were compared for further analysis. In most cases, samples that are misclassified by logistic regression algorithm are correctly classified by EBGWO-CatBoost algorithm. That is, the sample between good customers and bad customers, which the logistic regression algorithm cannot distinguish. In these samples, the values of important feature variables are not very prominent. Logistic regression algorithm is calculated according to the weight of these characteristic variables. Each characteristic variable is correlated with each other. Especially the important characteristic variables directly affect the result of the algorithm. So the logistic regression algorithm cannot distinguish between

them (the values of several important characteristic variables are not very prominent). CatBoost algorithm in EBGWO-CatBoost algorithm, the main method it uses when dealing with sample characteristic variables is the heat coding. It avoiding the existence of order or distance relationship algorithm. In the calculation process, each feature is encoded separately to weaken the relationship between features, thus reducing the influence of a single important feature on the whole algorithm. Therefore, when the value representation of important variable features is not very obvious, EBGWO-CatBoost algorithm can correctly classify them.

It has high accuracy, precise to distinguish good customers and bad customers, high classification performance and low algorithm complexity. Therefore, it is feasible to replace the logistic regression algorithm in the traditional risk control model with EBGWO-CatBoost algorithm.

## **6.8 Chapter Summary**

This chapter mainly introduces the GWO algorithm optimization process. Firstly, the GWO algorithm is optimized from the mathematical formula, and the optimized GWO algorithm (EBGWO) is coded. The new algorithm is then tested and compared on two different datasets. Finally, the combination of EBGWO and CatBoost algorithm to form a new algorithm (EBGWO-Catboost), and again through the dataset of experimental comparative analysis, finally concluded that the comprehensive performance of EBGWO-CatBoost algorithm is the best.

## CHAPTER 7

### CATBOOST EXPLAINABILITY OPTIMIZATION

#### 7.1 Overview

The SHAP (SHapley Additive exPlanations) framework is a powerful tool for interpreting machine learning models by quantifying the contribution of each feature to individual predictions. Drawing from cooperative game theory, SHAP values provide a detailed breakdown of how each feature affects the model's output, offering both positive and negative impacts.

In the chapter of the EBGWO-CatBoost algorithm applied to the China\_Credit and LendingClub datasets, SHAP values were utilized to enhance model explainability. For individual samples predicted as either "good" or "bad," SHAP values clearly illustrated how each feature influenced the prediction. In this study, two datasets, China\_Credit and LendingClub, were used for experiments. Individual samples from these datasets are analyzed in detail to understand the importance of their characteristics. In a single sample, the reasons for the significance of each feature are examined, and the impact of each feature's contribution on the prediction results is explained. In summary, SHAP provides a transparent view of feature impacts, enhances interpretability, and validates the alignment of model predictions with actual outcomes. This comprehensive approach allows for a deeper understanding of how features drive predictions, supporting better model insights and trust.

#### 7.2 SHAP framework

SHAP (SHapley Additive exPlanations) is a Python model interpretation package. Drawing inspiration from cooperative game theory, SHAP constructs an additive interpretation model (Rodríguez et al., 2022). It is based on Shapley values from cooperative

game theory, which allocate contributions fairly. SHAP helps us understand the decision process of models by calculating the contribution of each feature to the model's predictions.

The importance and advantages of SHAP include:

1. **Fairness and Consistency:** Shapley values provide a fair method for distributing each feature's contribution to the prediction outcome. This means each feature's contribution is calculated based on its average impact across all possible feature combinations.
2. **Global and Local Interpretability:** SHAP can explain individual predictions (local interpretability) and analyze the model's performance across the entire dataset (global interpretability). This helps in understanding and improving the model.
3. **Model Agnosticism:** SHAP can be applied to various types of models, including tree models (like XGBoost, LightGBM, CatBoost), linear models, and neural networks. This versatility ensures its usefulness across different machine learning tasks.
4. **Enhanced Interpretability:** SHAP values show the specific impact of each feature on predictions, increasing the model's transparency. This is particularly important in applications requiring high transparency, such as healthcare and finance.
5. **Visualization Tools:** SHAP provides various visualization tools, such as feature importance plots and partial dependence plots, which help users intuitively understand feature contributions.

Overall, the SHAP framework offers detailed explanations and intuitive visualizations, aiding users in understanding and trusting machine learning model predictions, especially with complex models and high-dimensional data.

For each prediction sample, the model produces a predictive value. The SHAP (SHapley Additive exPlanations) value is then assigned to each feature in the sample.

Assuming the  $j$ th feature value of the  $i$ -th sample is  $X_{i,j}$ , The SHAP value is shown in the following formula.

$$\hat{y}_i = \varphi_0 + \sum_{j=1}^m \varphi_{i,j} = \varphi_0 + \varphi_{i,1} \dots + \varphi_{i,m} \quad \text{Equation 7.1}$$

Where  $\varphi_0$  is the benchmark value of the model,  $\hat{y}_i$  is the predicted value of the  $i$ -th the sample, and  $\varphi_{i,j}$  is the Shapley values of  $X_{i,j}$ , indicating the contribution of the  $j$ -th feature in the  $i$ -th sample to the final predicted value. When  $\varphi_{i,j} > 0$ , it indicates that the feature enhances the predictive value of the model, indicating that the feature has a positive effect; When  $\varphi_{i,j} < 0$ , it indicates that the feature will reduce the predicted value, indicating that the feature has a negative effect. Compared to the importance of features, SHAP value compensates for the lack of explainability of feature importance, not only giving the importance level of feature quantity but also the positive and negative impact. The objective of SHAP (SHapley Additive exPlanations) is to provide explanations for the model's predictions by quantifying the impact of each feature variable on the prediction outcomes. The following will describe in detail the use of SHAP framework to improve the explainability of EBGWO-CatBoost algorithm.

### 7.3 Explainability experiment and discussion based on SHAP

Explanation of SHAP in the Context of CatBoost.

SHAP (Shapley Additive Explanations) is a method based on Shapley values from game theory that measures the contribution of each feature to the model's prediction. The core principle of SHAP is to decompose the model's output into the sum of each feature's contributions, ensuring that each feature's contribution to the model output is its fair share. SHAP calculates each feature value's contribution to the prediction by considering all

possible combinations of feature values. This process is based on two principles: fairness and local independence.

Fairness ensures that each feature's contribution is based on its actual impact on the model's output, making each feature's contribution equitable. Local independence simplifies the calculation by assuming that other feature values are independent when computing the contribution of a specific feature. SHAP values measure the marginal contribution of features and are one of the best methods for explaining models. SHAP can provide both global explanations (the impact of features on the overall model) and local explanations (the impact of features on individual predictions). It helps identify the main factors influencing individual predictions and their respective impact, which is particularly useful in risk management scenarios. For example, SHAP can explain why a model predicts a higher risk for a particular sample.

SHAP explains the principle of the CatBoost algorithm by calculating the marginal contribution of each feature to the model's output. The core idea of SHAP is to compute each feature's marginal contribution to the model's output, offering explanations for "black-box" models from both global and local perspectives. SHAP is an additive explanation model where all features are considered "contributors." For each prediction sample, the model generates a prediction value, and each feature of this sample has a corresponding SHAP value, representing the impact of that feature on the prediction. Some features positively influence the predicted probability, while others negatively affect it. By using SHAP, researchers can understand the logic behind model predictions, especially for complex models like CatBoost, revealing which categorical features have a significant impact on the prediction results. In the CatBoost algorithm, SHAP can analyze the impact of categorical features on the model's predictions. CatBoost is a gradient boosting algorithm library that

handles categorical features well, automatically processing and combining them to enrich the feature dimensions. SHAP helps to further understand how these categorical features influence the model's predictions, enhancing the interpretability and understanding of CatBoost's prediction logic.

### 7.3.1 Experimental discuss of China\_Credit dataset

This section will analyze the explainability characteristics of the experimental results of EBGWO-CatBoost algorithm in section 6.6.3, and use SHAP framework to further improve the explainability of EBGWO-CatBoost algorithm. Due to the large number of variable features, this study presents the Chi-square box details of the first three characteristic variables, as shown in Tables 7.1 - 7.3.

**Table 7.1: Chi square box of account balance**

| Number of boxes | Value range      | Sample number |
|-----------------|------------------|---------------|
| 1               | $(-\infty, 640]$ | 31020         |
| 2               | $(640, 680]$     | 20329         |
| 3               | $(680, +\infty]$ | 49277         |
| Total           | ——               | 100626        |

**Table 7.2: Chi square box of duration of credit (month)**

| Number of boxes | Value range       | Sample number |
|-----------------|-------------------|---------------|
| 1               | $(-\infty, 4650]$ | 10069         |
| 2               | $(4650, 9650]$    | 4804          |
| 3               | $(9650, 10300]$   | 10079         |

|       |                     |        |
|-------|---------------------|--------|
| 4     | (10300, 19300]      | 25154  |
| 5     | (19300, $+\infty$ ] | 52520  |
| Total | ——                  | 100626 |

**Table 7.3: Chi square box of length of current employment**

| Number of boxes | Value range         | Sample number |
|-----------------|---------------------|---------------|
| 1               | $(-\infty, 4500]$   | 1504          |
| 2               | (4500, 12200]       | 9841          |
| 3               | (1220, 23400]       | 35014         |
| 4               | (23400, 31100]      | 42358         |
| 5               | (31100, $+\infty$ ] | 11909         |
| Total           | ——                  | 100626        |

#### 7.3.1.1 SHAP framework analyzes individual sample processes

In this study, the experimental results of EBGWO-CatBoost algorithm in section 6.6.3 were used, that is, the "bad samples" and "good samples" predicted by EBGWO-CatBoost algorithm were selected for experimental verification. Select "bad sample" No. 3261 (with the highest probability of predicting bad samples) and "good sample" No. 7515 (with the highest probability of predicting good samples) as the control.

Through the experimental test of SHAP. 3261th sample value the details of the "bad sample" data are as follows in Table 7.4.

**Table 7.4: Examples of bad samples**

| Feature name                 | Original value | Value range        | Box number | SHAP value |
|------------------------------|----------------|--------------------|------------|------------|
| Account Balance              | 499            | $(-\infty, 640]$   | 1          | 3.3862     |
| Duration of Credit           | 35000          | $(19300, +\infty]$ | 5          | 0.4042     |
| Length of current employment | 98204          | $(31100, +\infty]$ | 5          | -0.0901    |
| ...                          | ...            | ...                | ...        | ...        |
| No of Credits at this Bank   | 16015<br>1     | $(15054, +\infty)$ | 5          | -0.0261    |
| Foreign Worker               | 0              | $[0,1]$            | 2          | 0.1804     |

A SHAP value is generated for each sample feature, where the baseline for the entire SHAP framework model is -1.1429. The specific SHAP interpretation process is as follows: Calculate the baseline  $\emptyset_0 = -1.1429$  and the SHAP values of 20 characteristic variables (SHAP values in Table 7.4) according to formula 7.1, and then add the baseline values with the SHAP values of each characteristic to get  $3.3415(-1.1429 + (3.3862 + 0.4042 \dots + 0.1804))$ , that is,  $\hat{y}_i = 4.4844$ , which is the predicted value of SHAP, and uses sigmoid function to convert it into probability (Bensic et al., 2005), that is  $P_i = \frac{1}{1 + e^{-\hat{y}_i}}$ , the result is greater than 0.5, so the prediction result is 1 (bad sample). The actual value of sample 3261 in the test set is 1 (bad sample), so the result of SHAP is consistent with the real sample value. The above process is summarized: the SHAP value of each characteristic variable in Table 6.14 reflects its contribution to the prediction result; Each characteristic variable pushes the model prediction from the baseline -1.1429 through the SHAP value to the final prediction of 3.3415, and finally converted into a probability value to determine whether it is a good or bad sample.

The simple understanding of the above calculation process is that the SHAP value of each feature variable reflects its contribution to the prediction result, and each feature variable pushes the prediction result of the SHAP framework model from the baseline " $\emptyset_0$ " to the final prediction result " $\hat{y}_i$ " through the SHAP value. If the value of " $\hat{y}_i$ " is greater than 0, then the result is closer to 1, and the final prediction sample is a "bad sample". If the value of " $\hat{y}_i$ " is less than 0, then the prediction result is closer to 0, and the final prediction sample is a "good sample".

After the above introduction, the following will take the above sample as an example to explain the model's prediction results for this sample in detail and dig deeply into how the SHAP framework works to give the prediction results.

First, SHAP's baseline of -0.1429,  $\emptyset_0$  has a value less than 0, that is, the initial prediction result is 0 (label 0-bit good sample), and the result classifies the sample as a good sample.

The SHAP value of the first feature variable Account Balance is 3.3862 ( $\hat{y}_i$ 's value), which pushes the prediction result of the model from -0.1429 to 3.2433 (-0.1429+3.3862), which is greater than 0, then, the calculated result is converted by Sigmoid function, and its probability value is finally greater than 0.5, so it is classified as a bad sample.

The SHAP value of the second characteristic variable Duration of Credit is 0.4042 ( $\hat{y}_i$ 's value), which pushes the prediction result of the model from 3.2433 to 3.6475 (3.2433+0.4042), and the result is greater than 0, then, the calculated result is converted by Sigmoid function, and its probability value is finally greater than 0.5, so it is classified as a bad sample.

And so on, the contribution of SHAP values of the other 20 characteristic variables to the prediction results of the model is considered in turn. The combined effect of 20 characteristic variables finally pushed the model's prediction result from the baseline -1.1429 to 3.3415 ( $\hat{y}_i$ 's value), and the result was greater than 0, then, the calculated result is converted by Sigmoid function, and its probability value is finally greater than 0.5, it is the bad sample. To sum up, 20 feature variables encourage the model to classify this sample as a bad sample, among which the feature variable Account Balance has the greatest contribution to such prediction. Visualize the above results as shown in Figure 7.1.

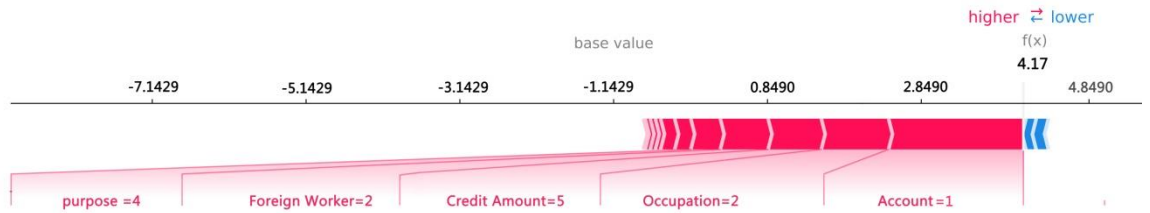


**Figure 7.1: Visualization of bad sample SHAP values**

The feature variables that push the SHAP model's predicted value to the top are represented in red, and the feature variables that push the model's predicted value to the bottom are represented in blue. Figure 7.1 shows that each feature has its own contribution, pushing the SHAP framework model's prediction from the base value of -1.1429 to the final value of 3.3415 ( $\hat{y}_i$ 's value). The conversion probability of 3.3415 is greater than 0.5, so the sample is a "bad sample". Since the experiment selected the sample of No. 3261 predicted by EBGWO-CatBoost algorithm, its real prediction result was "bad sample", which was consistent with the result predicted by SHAP framework. Therefore, the SHAP framework can explain the reason in detail any sample is identified as a good or bad sample, which greatly improves the interpretability of EBGWO-CatBoost algorithm.

As a contrast, through the experimental test of SHAP. 7515th sample value the details of the "bad sample" data are as follows in Table 7.5.

**Table 7.5: Good sample examples**

| Feature name                 | Original value | Value range        | Box number | SHAP value |
|------------------------------|----------------|--------------------|------------|------------|
| Account Balance              | 799            | $[680, +\infty)$   | 3          | -1.6066    |
| Duration of Credit           | 3000           | $(-\infty, 4650]$  | 1          | -0.9787    |
| Length of current employment | 19514          | $(1220, 23400]$    | 4          | -0.1078    |
| ...                          | ...            | ...                | ...        | ...        |
| No of Credits at this Bank   | 30354          | $(21581, +\infty)$ | 5          | -0.1471    |
| Foreign Worker               | 1              | $(6300, 10500]$    | 2          | -0.0353    |

The prediction process of the SHAP framework model for this sample is similar to that of the bad sample. 20 characteristic variables finally push the prediction result of the model from the baseline -1.1429 to -4.888 ( $\hat{y}_i$ 's value), and the result is less than 0, then, the calculated result is converted by Sigmoid function, and its probability value is finally less than 0.5, so it is classified as a good sample. The interpretation result given by the SHAP value is consistent with the classification result of the real value.

#### 7.3.1.2 SHAP framework analyzes whole sample processes

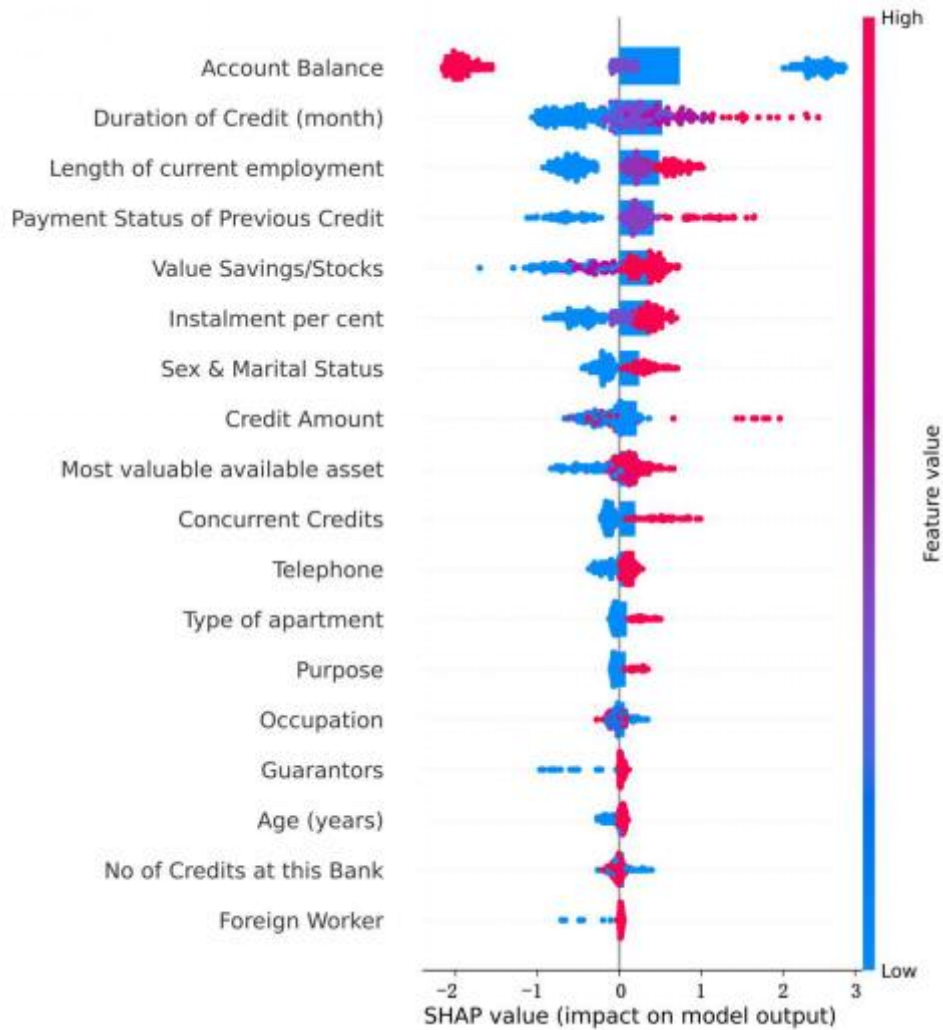


**Figure 7.2: Visualization of SHAP values for the overall sample**

Figure 7.2 is plotted from top to bottom in order of the importance of the features. Because there are so many features (over 100), some unimportant features, the SHAP framework defaults to not showing them in the graph. So, figures 7.2 show some of the important features. But these two graphs can be used to analyse all features. Because there are no features shown, the effect on the algorithm is very small.

As depicted in Figure 7.2, each row corresponds to a feature variable, with the horizontal axis representing the SHAP value. Each point on the plot represents a sample. The color intensity reflects the magnitude of the feature variable's value (Chi-square box numbers): redder colors indicate higher values, while bluer colors indicate lower values.

Note that the value here is not the original value of the feature variable, but the number of Chi square box in which its original value is located.

According to the analysis in Section 5.2.1, the characteristic variable Account Balance has the greatest impact on the model. Therefore, this section takes it as an example to introduce it in detail. As can be seen from Figure 7.2, for the characteristic variable Account Balance, the larger the number of Chi square box (3 boxes in Table 7.1), the SHAP value is Positive, it pushes this SHAP value higher, that is, causes the model to predict the sample closer to 1, and is more inclined to classify the sample as bad. And SHAP value are below -1, corresponding to the red part at the leftmost end of the first row in the figure. The smaller the number of Chi square box (1 boxes in Table 7.1), the SHAP value is positive, it pushes this SHAP value down, that is, causes the model to predict the sample closer to 0, and is more inclined to classify the sample as good. And SHAP value is between 2 and 3, corresponding to the blue part at the rightmost end of the first row in the figure; The number of Chi square box is in the middle (2 boxes in Table 7.1), most of the SHAP values are between 0 and 1, and a small part is less than 0, corresponding to the purple part in the middle of the first row in the figure.

Figure 7.3 shows the sample distribution of the feature "Account Balance" predicted by EBGWO-CatBoost algorithm.

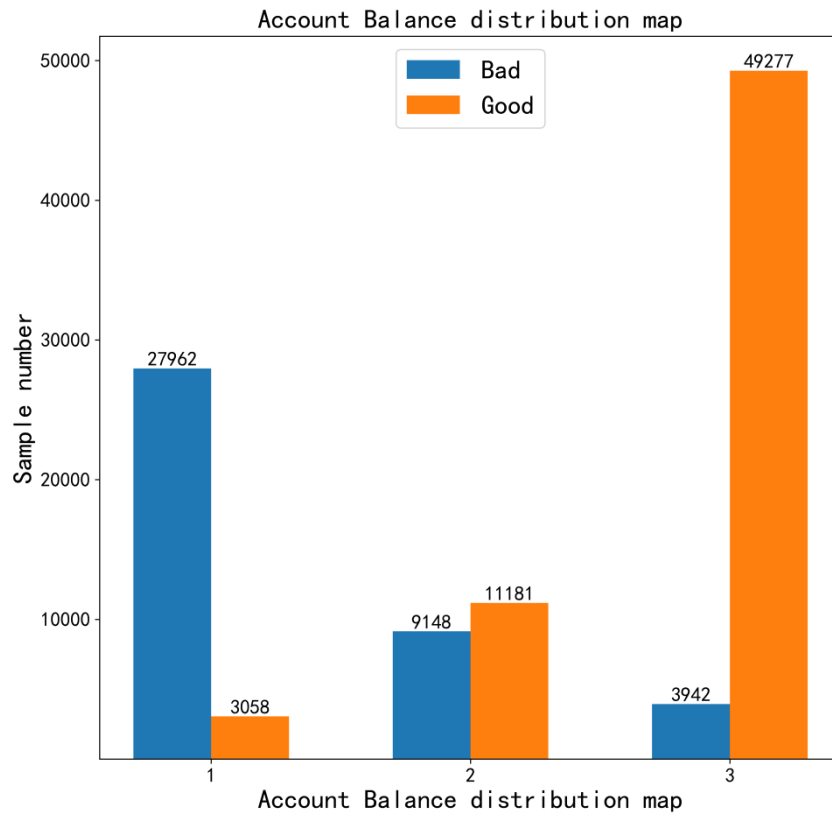


**Figure 7.3: Account Balance distribution (Classification of EBGWO-CatBoost)**

As explained in Figure 7.2 above, if the Chi square box is smaller, its SHAP value is positive (negative correlation and more bad samples), corresponding to the blue area on the right. As can be seen from the figure 7.3, the value of Account Balance is in the samples in box 1, 90% of which are bad samples. At this time, the Chi square box is the smallest, therefore, it corresponds to the blue region corresponding to the ordinate feature Account Balance at the horizontal value 2-3 in Figure 7.2, most of which are bad samples. Indicating that the predicted result under the SHAP framework is consistent with the actual value. Of the samples in box 2, 55% were good samples. More than 92% of the samples in box 3 are good samples. Also corresponds to figure 7.2. Therefore, the characteristic distribution of "Account Balance" tested by EBGWO-CatBoost algorithm is consistent with the interpretation given by SHAP interpretation method.

### 7.3.2 Experimental discuss of LendingClub dataset

This section will analyze the explainability characteristics of the experimental results of EBGWO-CatBoost algorithm in section 6.6.5, and use SHAP framework to further improve the explainability of EBGWO-CatBoost algorithm. In this section, as in section 7.3.1, the experiment of a single sample is carried out first. Due to a large number of variable features, this study chooses the "installment" features that have a relatively large impact on the algorithm to show. Table 7.6 is real distribution of "installment".

**Table 7.6: Chi square box of installment**

| Number of boxes | Value interval     | Sample number |
|-----------------|--------------------|---------------|
| 1               | $(-\infty, 604.3]$ | 17195         |
| 2               | $(604.3, 639.4]$   | 6394          |
| 3               | $(639.4, 674.5]$   | 12188         |
| 4               | $(674.5, 709.61]$  | 30860         |
| 5               | $(709.6, +\infty]$ | 61363         |
| Total           | ——                 | 128000        |

#### 7.3.2.1 SHAP framework analyzes individual sample processes

This section explains a single sample. The sample with the highest probability of being predicted as a bad sample in the test set is selected as the bad sample. Take the 52154 sample in the test set as the example of a "bad sample", the details of the sample data are as follows Table 7.7.

### Table 7.7: Examples of bad samples

| Feature name    | Original value | Value range        | Box number | SHAP value |
|-----------------|----------------|--------------------|------------|------------|
| total_pymnt_inv | 598            | $(-\infty, 604.3]$ | 2          | 0.097714   |
| installment     | 114            | $(86.2, 235.4]$    | 4          | -0.090405  |
| emp_length      | 651            | $(547.4, 2025]$    | 1          | -0.047768  |
| ...             | ...            | ...                | ...        | ...        |
| home_ownership  | 1584           | $(-\infty, 5627]$  | 5          | 0.050301   |
| purpose         | 2125           | $(709.6, +\infty]$ | 2          | -0.135530  |

The SHAP value of each characteristic variable pushed the model prediction from the baseline-1.303 to the final prediction 3.08( $3.08(-1.303+(-0.0977+0.090...-0.135530))$ ), and finally converted to the probability value (See section 7.3.1.1 for the conversion probability process), the result of calculation shows that is the bad sample. Visualize the above results as shown in Figure 7.4.



### Figure 7.4: Visualization of bad sample SHAP values

Figure 7.4 shows that each feature has its own contribution, pushing the SHAP framework model's prediction from the base value of -1.303 to the final value of 3.08. The feature variables that push the SHAP model's predicted value to the top are represented in

red, and the feature variables that push the model's predicted value to the bottom are represented in blue.

7.3.2.2 Analysis of the whole sample

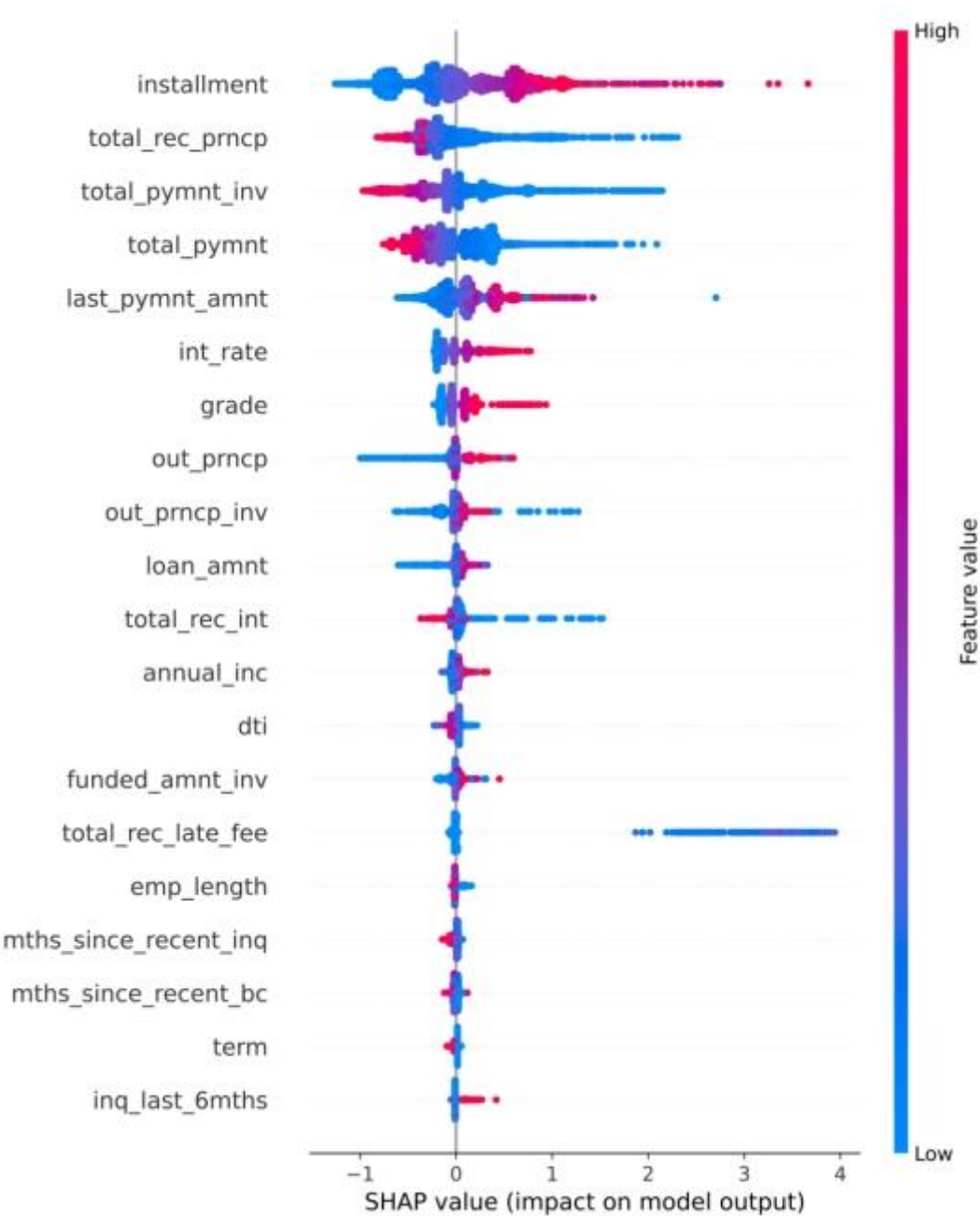


Figure 7.5: Visualization of SHAP values for the overall sample

Figure 7.5 is plotted from top to bottom in order of the importance of the features. Because there are so many features (over 100), some unimportant features, the SHAP framework defaults to not showing them in the graph. So, figures 7.5 show some of the important features. But these two graphs can be used to analyse all features. Because there are no features shown, the effect on the algorithm is very small.

In Figure 7.5, each row represents a feature, with the horizontal axis representing the SHAP value. Each point on the plot corresponds to a sample. It can be observed that the feature variable "installment" exhibits the highest importance, as indicated by its prominence in the plot. Sotakethis characteristic variable as an example. The "installment" Chi square box situation as shown in Table 7.8.

**Table 7.8: Chi square box of installment situation**

| Number of boxes | Value interval     | Sample number |
|-----------------|--------------------|---------------|
| 1               | $(-\infty, 604.3]$ | 17195         |
| 2               | $(604.3, 639.4]$   | 6394          |
| 3               | $(639.4, 674.5]$   | 12188         |
| 4               | $(674.5, 709.61]$  | 30860         |
| 5               | $(709.6, +\infty]$ | 61363         |
| Total           | ——                 | 128000        |

According to Figure 7.6, this figure is the classification prediction result of the EBGWO-CatBoost model, and the figure shows the situation of the feature "installment". The value of "installment" indicates that 96.6 percent of the time samples in the first Chi square box are bad samples; According to the content analysis in Section 7.3.2.2, if the Chi square box is smaller, the corresponding SHAP value in Figure 7.5 is positive, and its value

ranges from 3 to 4, then this sample is predicted to be a "bad sample"(see section 7.3.1.1). The actual classification results of the EBGWO-CatBoost algorithm are also mostly bad samples, accounting for 96.6%. Therefore, the predicted situation under the SHAP framework is consistent with the actual classification of the EBGWO-CatBoost algorithm. 76.6% in the second box are bad samples, 33.8% in the third box are bad samples, 76.6% in the second box are bad samples, 33.8% in the third box are bad samples, 83.6% in the fourth box are good samples, and 93.3% in the fifth box are good samples. And 93.3% in the fifth box are good samples. The predicted results of these Chi square box numbers under the SHAP framework are also consistent with the actual classification of the EBGWO-CatBoost algorithm.

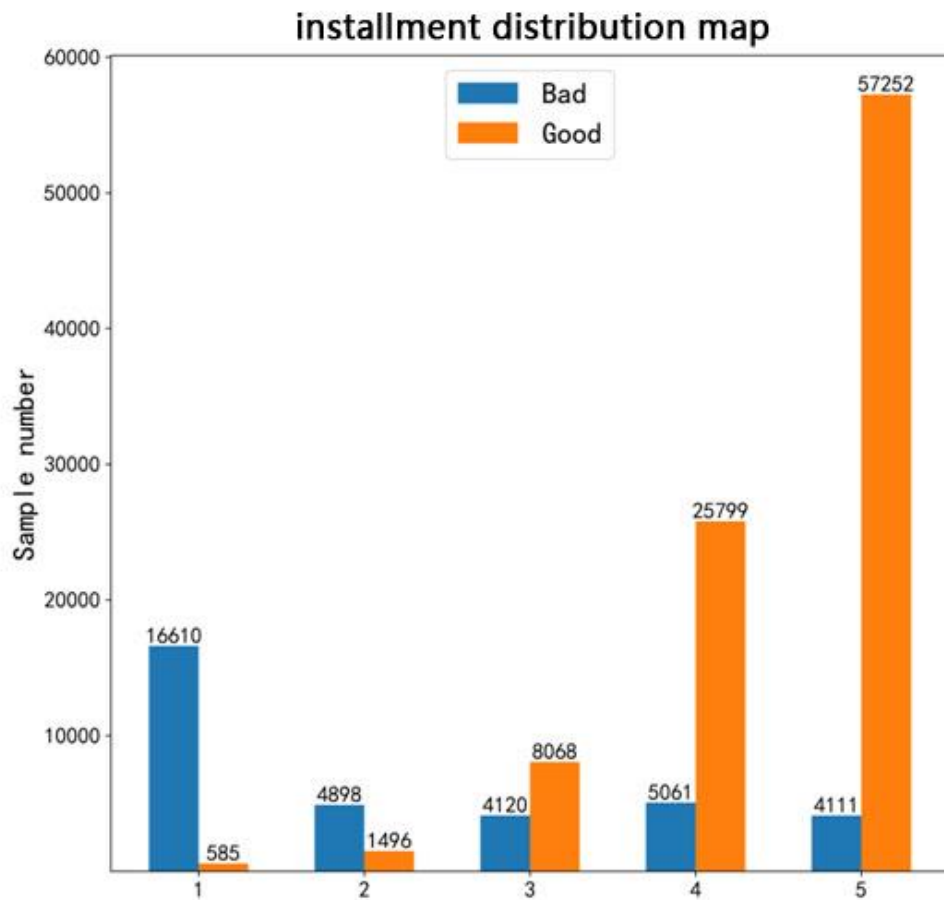


**Figure 7.6: Installment distribution (Classification of EBGWO-CatBoost)**

### **7.3.3 Conclusion**

Through SHAP framework, the interpretability of EBGWO-CatBoost algorithm can be further improved. After the classification of EBGWO-CatBoost algorithm, it is not possible to explain in detail why the algorithm classifies in this way, while the SHAP framework can explain the data classification process in detail. Whether the whole sample or a single sample can be explained.

## **7.4 Chapter Summary**

In this chapter, SHAP framework is used to explain the prediction classification results of EBGWO-CatBoost algorithm, which solves the problem that EBGWO-CatBoost algorithm is insufficient in the interpretation of features.

## CHAPTER 8

### CONCLUSION AND CONTRIBUTION

#### 8.1 Conclusion

In the process of selecting algorithms for constructing risk control models, the traditional choice has been the logistic regression algorithm. Logistic regression is favored due to its simplicity, rapid execution, and strong interpretability, making it a popular choice in risk control applications. However, as financial risk loan data has increased, the disparity between good and bad customers has become more pronounced. The traditional logistic regression algorithm, while effective in many scenarios, often struggles with low overall accuracy and limited capability in distinguishing between good and bad customers. This limitation underscores the need to optimize traditional risk control models and explore new algorithms to address these challenges more effectively.

In this study, a hybrid algorithm is employed to tackle the issues of low accuracy in traditional risk control models and the limited ability to differentiate between good and bad customers. The rationale behind utilizing hybrid algorithms is to leverage the strengths of various algorithms, aiming to enhance the overall performance of the risk control model. By combining different algorithms, it becomes possible to capitalize on their respective advantages and address the shortcomings of individual methods, ultimately leading to a more effective and robust risk control solution. Solving the risk control model is the key to improving model accuracy and the ability to distinguish between good and bad customers. In order to solve these two problems, this study chose the three algorithms with the highest accuracy: LightGBM, XGBoost, and CatBoost. The study selected the best comprehensive performance CatBoost algorithm through experimental comparison. The CatBoost algorithm

improves the accuracy of risk control models by over 15%. However, the CatBoost algorithm also has some disadvantages compared with the logistic regression algorithm. The biggest disadvantages are three aspects: First, the classification performance of the CatBoost algorithm is lower than that of the logistic regression algorithm. Second, it is difficult for the CatBoost algorithm to find the optimal parameter. To obtain a higher accuracy rate, one needs to find the optimal parameter manually. This way of finding the parameter is very complicated, which seriously increases the complexity of the algorithm. The third is the weakness of the algorithm's interpretability. Therefore, solving the defects of the CatBoost algorithm is the key to improving the comprehensive performance of the risk control model.

In view of the low classification performance question of the CatBoost algorithm, corresponding to the first question raised in section 1.4, the model has high accuracy and good and bad customer identification ability, but it also has low classification performance. This study, based on the literature review in Chapter 2, identifies that the classification performance of the CatBoost algorithm is greatly affected by the number of features in the dataset. When the number of features in the dataset is particularly large, the classification performance of the risk control model established by the CatBoost algorithm will be very low, resulting in long running times. Therefore, this study first calculated the IV (Information Value) value in the dataset and then screened out the variable features with a value less than 0.03, indicating low correlation and very little impact on the model. The study then recalculated and verified these features using Mutual Information. The features with low correlation degree (those that have little influence on the algorithm) were deleted. Finally, the data with deleted features was put into the CatBoost algorithm for the experiment. The experimental results show that the running time of the CatBoost algorithm improved by about 20% (indicating improved classification performance) compared with that of the

traditional risk control model algorithm under the condition of no or very small impact on variables, which is not much different from that of the traditional risk control model algorithm. Therefore, the improvement in algorithm classification performance can meet the needs of the risk control model. This achievement aligns with the goals proposed in section 1.5, which are to improve the traditional risk control model algorithm by replacing the original algorithm (logistic regression algorithm) with a more advanced algorithm (CatBoost algorithm) and to improve model accuracy and the ability to identify good and bad customers. By using dimensionality reduction (variable selection), the operation classification performance of the algorithm was enhanced.

In view of the high complexity question of the CatBoost algorithm, corresponding to the second problem raised in section 1.4, the study addresses how to further improve the accuracy of the model while reducing the complexity of the risk control model. This study proposed an improved CatBoost algorithm (EBGWO-CatBoost), which combines the improved GWO algorithm (EBGWO) with the CatBoost algorithm. The optimized GWO algorithm was used to offset the defects of the CatBoost algorithm in parameter tuning. The EBGWO-CatBoost combined algorithm model is divided into four steps. In the first step, the advantages of the Grey Wolf algorithm (GWO) are utilized to find the optimal parameters, and the optimal parameters of the CatBoost algorithm are determined to reduce the use of computing resources and complexity of the model. However, the balance between the exploration and exploitation capabilities of the GWO algorithm itself needs to be further optimized. Therefore, the second step requires improving the GWO algorithm, referred to as the "EBGWO" algorithm. The EBGWO algorithm is an enhanced version of the Grey Wolf Optimization algorithm, where the number of leading wolves in the algorithm changes from a fixed 3 to 2 or 4. The optimized GWO algorithm (EBGWO algorithm) dynamically selects

the number of leading wolves according to the data characteristics of the dataset, which not only improves the classification performance of the algorithm but also effectively balances the exploration and exploitation capabilities, avoiding the algorithm from falling into local optimal solutions. This improvement enhances predictive accuracy and simplifies algorithm complexity. The third step involves applying the new algorithm to the risk control model for testing and comparison. The model established by the EBGWO-CatBoost algorithm exhibits more advantages compared to models built by other algorithms. The objectives set out in section 1.5 have been achieved, which include using the GWO algorithm to automatically find the optimal parameters of the CatBoost algorithm, simplifying the complexity of the CatBoost algorithm. Addressing the shortcomings of the GWO algorithm, the new EBGWO algorithm prevents the GWO algorithm from falling into local optimal solutions, balances the exploration and exploitation abilities of the algorithm, and improves the accuracy of the CatBoost algorithm by about 4%. Compared with the logistic regression algorithm, the accuracy of the new optimized algorithm improved by about 20%.

In view of the weak interpretability question of the CatBoost algorithm, corresponding to the third problem raised in section 1.4, the study addresses how to improve the explainability of the risk control model. Explainability is crucial for risk control models because they need not only to accurately distinguish between good and bad customers but also to explain why the model is classified in a particular way. The CatBoost algorithm itself has some interpretability but is insufficient compared to logistic regression, requiring further improvement. This study proposes using the SHAP framework to calculate the contribution value of each characteristic variable in each sample to enhance the interpretability of the CatBoost algorithm. First, the study shows the number of Chi-square boxes for each feature variable in the sample. Then, the SHAP values of each characteristic variable of the sample

are calculated, and the SHAP values of all characteristic variables are added together and converted into probability values to determine whether the sample is a good sample (good customer) or a bad sample (bad customer). Finally, the classification results (good customers or bad customers) under the SHAP framework are compared with the classification results of the EBGWO-CatBoost algorithm. If the two results match, it indicates that the SHAP framework correctly explains why the model algorithm is classified in this way, thus meeting the requirements of the risk control model.

## **8.2 Research Contributions**

In view of the weak interpretability question of the CatBoost algorithm, corresponding to the third problem raised in section 1.4, the study addresses how to improve the explainability of the risk control model. Explainability is crucial for risk control models because they need not only to accurately distinguish between good and bad customers but also to explain why the model is classified in a particular way. The CatBoost algorithm itself has some interpretability but is insufficient compared to logistic regression, requiring further improvement. This study proposes using the SHAP framework to calculate the contribution value of each characteristic variable in each sample to enhance the interpretability of the CatBoost algorithm. First, the study shows the number of Chi-square boxes for each feature variable in the sample. Then, the SHAP values of each characteristic variable of the sample are calculated, and the SHAP values of all characteristic variables are added together and converted into probability values to determine whether the sample is a good sample (good customer) or a bad sample (bad customer). Finally, the classification results (good customers or bad customers) under the SHAP framework are compared with the classification results of the EBGWO-CatBoost algorithm. If the two results match, it indicates that the SHAP

framework correctly explains why the model algorithm is classified in this way, thus meeting the requirements of the risk control model.

### **8.3 Recommendation for Future Works**

Detailed Analysis and Comparative Study: To further enhance the understanding and efficacy of the proposed EBGWO-CatBoost model, a detailed analysis of misclassifications could provide valuable insights. Specifically, investigating profiles that are misclassified by logistic regression but correctly classified by the EBGWO-CatBoost model—and vice versa—could highlight the strengths and limitations of each approach.

This analysis could be conducted as follows: 1. Detailed Misclassification Analysis: Profiles Misclassified by Logistic Regression but Correctly Classified by EBGWO-CatBoost. Identify Profiles: Select a representative set of profiles (both good and bad customers) that were misclassified by logistic regression but correctly classified by the EBGWO-CatBoost model. Analyze Misclassification: Use the SHAP framework to analyze why the EBGWO-CatBoost model correctly classified these profiles. SHAP values can show which features were most influential in the correct classification, helping to understand the model's decision-making process. Compare with Logistic Regression: Examine the features and decision boundaries used by logistic regression for these profiles. Determine if there are specific features or interactions that logistic regression fails to capture, which the EBGWO-CatBoost model can handle effectively. 2. Profiles Misclassified by EBGWO-CatBoost but Correctly Classified by Logistic Regression. Identify Profiles: Select a set of profiles that were correctly classified by logistic regression but misclassified by the EBGWO-CatBoost model. Analyze Misclassification: Utilize SHAP values to explore why the EBGWO-CatBoost model misclassified these profiles. Determine which features or aspects of the data might have led to the misclassification. Compare with EBGWO-CatBoost:

Assess the decision-making process of logistic regression for these profiles. Identify if there are any weaknesses or limitations in the EBGWO-CatBoost model that could be addressed.

**Exploration of Feature Importance and Model Behavior:** **Feature Importance Analysis:** Compare the feature importance scores of both models. Determine if there are significant differences in how each model values certain features and how this impacts classification accuracy. **Model Decision Boundaries:** Visualize and compare the decision boundaries of logistic regression and EBGWO-CatBoost models. Understanding these boundaries can provide insights into why certain profiles are misclassified by one model but not the other.

**Potential Future Research Directions:** **1. Enhancing Model Robustness.** **Feature Engineering:** Based on the misclassification analysis, develop new features or modify existing ones to address weaknesses identified in the EBGWO-CatBoost model. **Hybrid Approaches:** Explore hybrid models that combine strengths from both logistic regression and EBGWO-CatBoost, potentially using ensemble methods to leverage the best of both approaches. **2. Improved Interpretability Techniques.** **Advanced Interpretability Methods:** Investigate other interpretability techniques alongside SHAP, such as LIME (Local Interpretable Model-agnostic Explanations), to gain further insights into model decisions. **Model Explanation Frameworks:** Develop frameworks that provide a more comprehensive explanation of why certain profiles are misclassified, incorporating both global and local interpretability methods. **3. Model Tuning and Optimization.** **Advanced Interpretability Methods:** Investigate other interpretability techniques alongside SHAP, such as LIME (Local Interpretable Model-agnostic Explanations), to gain further insights into model decisions. **Model Explanation Frameworks:** Develop frameworks that provide a more comprehensive explanation of why certain profiles are misclassified, incorporating both global and local interpretability methods.

## REFERENCES

- Asha, R. B., & KR, S. K. (2021). Credit card fraud detection using artificial neural network. *Global Transitions Proceedings*, 2(1), 35-41.
- Anjali, M., Dubey, M. S., Shivhare, M. R., & Dixit, M. M. (2018). Journal Of Harmonized Research (JOHR). *Journal Of Harmonized Research in Applied Sciences*, 6(4), 276-285.
- Ahmed, R., Rangaiah, G. P., Mahadzir, S., Mirjalili, S., Hassan, M. H., & Kamel, S. (2023). Memory, evolutionary operator, and local search based improved Grey Wolf Optimizer with linear population size reduction technique. *Knowledge-Based Systems*, 264, 110297.
- Arora, S., Singh, H., Sharma, M., Sharma, S., & Anand, P. (2019). A new hybrid algorithm based on grey wolf optimization and crow search algorithm for unconstrained function optimization and feature selection. *Ieee Access*, 7, 26343-26361.
- Abdulhafedh, A. (2022). Comparison between common statistical modeling techniques used in research, including: Discriminant analysis vs logistic regression, ridge regression vs LASSO, and decision tree vs random forest. *Open Access Library Journal*, 9(2), 1-19.
- Addy, W. A., Ajayi-Nifise, A. O., Bello, B. G., Tula, S. T., Odeyemi, O., & Falaiye, T. (2024). Machine learning in financial markets: A critical review of algorithmic trading and risk management. *International Journal of Science and Research Archive*, 11(1), 1853-1862.

- Ahn, J. M., Kim, J., & Kim, K. (2023). Ensemble machine learning of gradient boosting (XGBoost, LightGBM, CatBoost) and attention-based CNN-LSTM for harmful algal blooms forecasting. *Toxins*, 15(10), 608.
- Ajmal, M. M., Khan, M., & Shad, M. K. (2021). The global economic cost of coronavirus pandemic: current and future implications. *Public Administration and Policy*, 24(3), 290-305.
- Bücker, M., Szepannek, G., Gosiewska, A., & Biecek, P. (2022). Transparency, auditability, and explainability of machine learning models in credit scoring. *Journal of the Operational Research Society*, 73(1), 70-90.
- Blanco, A., Pino-Mejías, R., Lara, J., & Rayo, S. (2013). Credit scoring models for the microfinance industry using neural networks: Evidence from Peru. *Expert Systems with applications*, 40(1), 356-364.
- Bentéjac, C., Csörgő, A., & Martínez-Muñoz, G. (2021). A comparative analysis of gradient boosting algorithms. *Artificial Intelligence Review*, 54, 1937-1967.
- Bolton, J. M., Pagura, J., Enns, M. W., Grant, B., & Sareen, J. (2010). A population-based longitudinal study of risk factors for suicide attempts in major depressive disorder. *Journal of psychiatric research*, 44(13), 817-826.
- Bekhet, H. A., & Eletter, S. F. K. (2014). Credit risk assessment model for Jordanian commercial banks: Neural scoring approach. *Review of Development Finance*, 4(1), 20-28.
- Berkson, J. (1944). Application of the logistic function to bio-assay. *Journal of the American statistical association*, 39(227), 357-365.

- Bensic, M., Sarlija, N., & Zekic - Susac, M. (2005). Modelling small - business credit scoring by using logistic regression, neural networks and decision trees. *Intelligent Systems in Accounting, Finance & Management: International Journal*, 13(3), 133-150.
- Brentnall, A. R., Cuzick, J., Buist, D. S., & Bowles, E. J. A. (2018). Long-term accuracy of breast cancer risk assessment combining classic risk factors and breast density. *JAMA oncology*, 4(9), e180174-e180174.
- Bhattacharya, A., Biswas, S. K., & Mandal, A. (2023). Credit risk evaluation: a comprehensive study. *Multimedia Tools and Applications*, 82(12), 18217-18267.
- Chang, Y. C., Chang, K. H., & Wu, G. J. (2018). Application of eXtreme gradient boosting trees in the construction of credit risk assessment models for financial institutions. *Applied Soft Computing*, 73, 914-920.
- Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 785-794).
- Chongren, W., & Dongmei, H. (2019). Personal credit evaluation of Internet credit based on hyperparameter optimization and integrated learning [J]. *Statistics and decision-making*, 35(01), 89-93.
- Coşkun, S. B., & Turanlı, M. (2023). Credit risk analysis using boosting methods. *Journal of Applied Mathematics, Statistics and Informatics*, 19(1), 5-18.
- Chen, H., Li, C., Mafarja, M., Heidari, A. A., Chen, Y., & Cai, Z. (2023). Slime mould algorithm: A comprehensive review of recent variants and applications. *International Journal of Systems Science*, 54(1), 204-235.

- Das, S., Khan, M. A. A., Mahmoud, E. E., Abdel-Aty, A. H., Abualnaja, K. M., & Shaikh, A. A. (2021). A production inventory model with partial trade credit policy and reliability. *Alexandria Engineering Journal*, 60(1), 1325-1338.
- Dinca, G., & Bociu, M. (2015). Using discriminant analysis for credit decision. Bulletin of the Transilvania University of Brasov. *Series V: Economic Sciences*, 277-288.
- Devika, M., Kishan, S. R., Manohar, L. S., & Vijaya, N. (2022, December). Credit Card Fraud Detection Using Logistic Regression. In *2022 Second International Conference on Advanced Technologies in Intelligent Control, Environment, Computing & Communication Engineering (ICATIECE)* (pp. 1-6).
- Demir, S., & Sahin, E. K. (2023). Predicting occurrence of liquefaction-induced lateral spreading using gradient boosting algorithms integrated with particle swarm optimization: PSO-XGBoost, PSO-LightGBM, and PSO-CatBoost. *Acta Geotechnica*, 18(6), 3403-3419.
- Dina, A. S., Siddique, A. B., & Manivannan, D. (2022). Effect of balancing data using synthetic data on the performance of machine learning classifiers for intrusion detection in computer networks. *IEEE Access*, 10, 96731-96747.
- Dhavakumar, P., & Gopalan, N. P. (2021). An efficient parameter optimization of software reliability growth model by using chaotic grey wolf optimization algorithm. *Journal of Ambient Intelligence and Humanized Computing*, 12, 3177-3188.
- Demir, S., & Sahin, E. K. (2023). Predicting occurrence of liquefaction-induced lateral spreading using gradient boosting algorithms integrated with particle swarm optimization: PSO-XGBoost, PSO-LightGBM, and PSO-CatBoost. *Acta Geotechnica*, 18(6), 3403-3419.

- Esenogho, E., Mienye, I. D., Swart, T. G., Aruleba, K., & Obaido, G. (2022). A neural network ensemble with feature engineering for improved credit card fraud detection. *IEEE Access*, 10, 16400-16407.
- Edunjobi, T. E., & Odejide, O. A. (2024). Theoretical frameworks in AI for credit risk assessment: Towards banking efficiency and accuracy. *International Journal of Scientific Research Updates 2024*, 7(01), 092-102. (2.2)
- Edunjobi, T. E., & Odejide, O. A. (2024). Theoretical frameworks in AI for credit risk assessment: Towards banking efficiency and accuracy. *International Journal of Scientific Research Updates 2024*, 7(01), 092-102.
- Fernandes, G. B., & Artes, R. (2016). Spatial dependence in credit risk and its improvement in credit scoring. *European Journal of Operational Research*, 249(2), 517-524.
- Fogarty, D. J. (2006). Multiple imputation as a missing data approach to reject inference on consumer credit scoring. *Interstat*, 41(1), 41.
- Gürsoy, G., & Varol, A. (2021, December). Risks of Digital Transformation: Review of Machine Learning Algorithms in Credit Card Fraud Detection. In *2021 2nd International Informatics and Software Engineering Conference (IISEC)* (pp. 1-6).
- Gamini, P., Yerramsetti, S. T., Darapu, G. D., Pentakoti, V. K., & Raju, V. P. (2021). Detection of credit card fraudulent transactions using boosting algorithms. *Journal of Emerging Technologies and Innovative Research (JETIR)*, 8(2), 2021.
- Gao, B., Guan, H., Shen, W., & Ye, Y. (2022). Application of the gray wolf optimization algorithm in active disturbance rejection control parameter tuning of an electro-hydraulic servo unit. *Machines*, 10(8), 599.

- Hu, P., Chen, S., Huang, H., Zhang, G., & Liu, L. (2018). Improved alpha-guided grey wolf optimizer. *IEEE Access*, 7, 5421-5437.
- Hu, J., Chen, H., Heidari, A. A., Wang, M., Zhang, X., Chen, Y., & Pan, Z. (2021). Orthogonal learning covariance matrix for defects of grey wolf optimizer: Insights, balance, diversity, and feature selection. *Knowledge-Based Systems*, 213, 106684.
- Hatta, N. M., Zain, A. M., Sallehuddin, R., Shayfull, Z., & Yusoff, Y. (2019). Recent studies on optimisation method of Grey Wolf Optimiser (GWO): a review (2014–2017). *Artificial intelligence review*, 52, 2651-2683.
- Heidari, A. A., Abbaspour, R. A., & Chen, H. (2019). Efficient boosted grey wolf optimizers for global search and kernel extreme learning machine training. *Applied Soft Computing*, 81, 105521.
- Hasan, M. M., Yajuan, L., & Khan, S. (2022). Promoting China's inclusive finance through digital financial services. *Global Business Review*, 23(4), 984-1006.
- Islam, M. B., Islam, K. S., Khan, M. H., Al Omari, A. M., & Hasibunnahar, S. (2022, September). Detect deception on banking credit card payment system by machine learning classifiers. In *Second International Conference on Cloud Computing and Mechatronic Engineering (I3CME 2022)* (Vol. 12339, pp. 506-516).
- Ibrahim, R. A., Abd Elaziz, M., & Lu, S. (2018). Chaotic opposition-based grey-wolf optimization algorithm based on differential evolution and disruption operator for global optimization. *Expert systems with applications*, 108, 1-27.
- Ignatieva, K., & Trück, S. (2016). Modeling spot price dependence in Australian electricity markets with applications to risk management. *Computers & Operations Research*, 66, 415-433.

- Jacques, O. (2023). Explaining willingness to pay taxes: The role of income, education, ideology. *Journal of European Social Policy*, 33(3), 267-284.
- Kaastra, I., & Boyd, M. (1996). Designing a neural network for forecasting financial and economic time series. *Neurocomputing*, 10(3), 215-236.
- Khemakhem, S., & Boujelbene, Y. (2015). Credit risk prediction: A comparative study between discriminant analysis and the neural network approach. *Accounting and Management Information Systems*, 14(1), 60.
- Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... & Liu, T. Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30.
- Karimi, F., & Vaez, S. R. H. (2019). Two-stage optimal seismic design of steel moment frames using the LRFD-PBD method. *Journal of Constructional Steel Research*, 155, 77-89.
- Kinnander, M. (2020). Predicting profitability of new customers using gradient boosting tree models: Evaluating the predictive capabilities of the XGBoost, LightGBM and CatBoost algorithms. *Digitala Vetenskapliga Arkivet*, 2(1), 23-34.
- Karimi, F., & Vaez, S. R. H. (2019). Two-stage optimal seismic design of steel moment frames using the LRFD-PBD method. *Journal of Constructional Steel Research*, 155, 77-89.
- Liang, Z., Du, J., Hua, Y., Si, Y., & Li, M. (2023). Research on credit evaluation indicator system of high-tech SMEs: from the social capital perspective. *Systems*, 11(3), 141.
- Liang, W., Luo, S., Zhao, G., & Wu, H. (2020). Predicting hard rock pillar stability using GBDT, XGBoost, and LightGBM algorithms. *Mathematics*, 8(5), 765.

- Langley, P., & Sage, S. (1994, July). Oblivious decision trees and abstract cases. *In Working notes of the AAAI-94 workshop on case-based reasoning* (pp. 113-117).
- Li, Y., & Liu, Z. (2022). Performance Evaluation of Precision Poverty Alleviation with the Assistance of Wireless Communication Technology for Rural Revitalization. *Wireless Communications and Mobile Computing*, 2022(1), 4621121.
- Li Yijuan & Li Zhi. (2023). CGWO UAV path planning based on improved convergence factor. *Science and Technology Innovation and Application* (13),75-78.
- Li, T., Sun, D., Jing, P., & Yang, K. (2018). Smart card data mining of public transport destination: A literature review. *Information*, 9(1), 18.
- Liang, W., Luo, S., Zhao, G., & Wu, H. (2020). Predicting hard rock pillar stability using GBDT, XGBoost, and LightGBM algorithms. *Mathematics*, 8(5), 765.
- Li, Y., Mai, Y., Lin, Z., & Liang, S. (2020, July). Online transaction detection method using catboost model. *In 2020 International Conference on Communications, Information System and Computer Engineering (CISCE)* (pp. 236-240).
- Long, W., Jiao, J., Liang, X., Cai, S., & Xu, M. (2019). A random opposition-based learning grey wolf optimizer. *IEEE access*, 7, 113810-113825.
- Li, S., Zhang, H., Ma, R., Zhou, J., Wen, J., & Zhang, B. (2023). Linear discriminant analysis with generalized kernel constraint for robust image classification. *Pattern Recognition*, 136, 109196.
- Ma, X., Sha, J., Wang, D., Yu, Y., Yang, Q., & Niu, X. (2018). Study on a prediction of P2P network loan default based on the machine learning LightGBM and XGboost algorithms according to different high dimensional data cleaning. *Electronic Commerce Research and Applications*, 31, 24-39.

- Ma, X., Sha, J., Wang, D., Yu, Y., Yang, Q., & Niu, X. (2018). Study on a prediction of P2P network loan default based on the machine learning LightGBM and XGboost algorithms according to different high dimensional data cleaning. *Electronic Commerce Research and Applications*, 31, 24-39.
- Martín-Baos, J. Á., Garcia-Rodenas, R., & Rodriguez-Benitez, L. (2021). Revisiting kernel logistic regression under the random utility models perspective. An interpretable machine-learning approach. *Transportation Letters*, 13(3), 151-162.
- Masmoudi, K., Abid, L., & Masmoudi, A. (2019). Credit risk modeling using Bayesian network with a latent variable. *Expert Systems with Applications*, 127, 157-166.
- Mohammad, A. H. (2021). Intrusion detection using a new hybrid feature selection model. *Intelligent Automation & Soft Computing*, 30(1), 10-11.
- Mahdad, B., & Srairi, K. (2018). Blackout risk prevention in a smart grid based flexible optimal strategy using Grey Wolf-pattern search algorithms. *Energy Conversion and Management*, 98, 411-429.
- Mushava, J., & Murray, M. (2022). A novel XGBoost extension for credit scoring class-imbalanced data combining a generalized extreme value link and a modified focal loss function. *Expert Systems with Applications*, 202, 117233.
- Misheva, B. H., Osterrieder, J., Hirs, A., Kulkarni, O., & Lin, S. F. (2021). Explainable AI in credit risk management. *arXiv preprint arXiv: 2103.00949*.
- Marom, N. D., Rokach, L., & Shmilogici, A. (2010, November). Using the confusion matrix for improving ensemble classifiers. *In 2010 IEEE 26-th Convention of Electrical and Electronics Engineers in Israel* (pp. 000555-000559).

- Minh, H. L., Sang-To, T., Khatir, S., Wahab, M. A., & Cuong-Le, T. (2023). Damage identification in high-rise concrete structures using a bio-inspired meta-heuristic optimization algorithm. *Advances in Engineering Software*, 176, 103399.
- Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in engineering software*, 69, 46-61.
- Matteucci, F., Arzamasov, V., & Böhm, K. (2024). A benchmark of categorical encoders for binary classification. *Advances in Neural Information Processing Systems*, 36.
- Noh, H. J., Roh, T. H., & Han, I. (2005). Prognostic personal credit risk model considering censored information. *Expert Systems with Applications*, 28(4), 753-762.
- Nawai, N., & Shariff, M. N. M. (2010). Determinants of repayment performance in microcredit programs: A review of literature. *International Journal of Business and Social Science*, 1(2), 1-2.
- Nasiri, J., & Khiyabani, F. M. (2018). A whale optimization algorithm (WOA) approach for clustering. *Cogent Mathematics & Statistics*, 5(1), 1483565.
- Orgad, L., & Reijers, W. (2021). How to Make the Perfect Citizen? Lessons from China's Social Credit System. *Vand. J. Transnat'l L.*, 54, 1087.
- Ponsam, J. G., Gracia, S. J. B., Geetha, G., Karpaselvi, S., & Nimala, K. (2021, December). Credit Risk Analysis using LightGBM and a comparative study of popular algorithms. In *2021 4th International Conference on Computing and Communications Technologies (ICCCCT)* (pp. 634-641).
- Precup, R. E., David, R. C., & Petriu, E. M. (2016). Grey wolf optimizer algorithm-based tuning of fuzzy control systems with reduced parametric sensitivity. *IEEE Transactions on Industrial Electronics*, 64(1), 527-534.

- Paul, K., & Basu, D. (2024). Data Expedition: Travel Through Data Preprocessing, EDA And PCA. *Educational Administration: Theory and Practice*, 30(6), 2576-2590.
- Qi, J., Yang, R., & Wang, P. (2021, June). Application of explainable machine learning based on Catboost in credit scoring. *In Journal of Physics: Conference Series* (Vol. 1955, No. 1, p. 012039).
- Qu, Q., Wang, H. H., & Qi, M. L. (2022). Improved Grey Wolf Optimizer with Differential Perturbation for Function Optimization. *IAENG International Journal of Applied Mathematics*, 52(2).
- Qi, J., Yang, R., & Wang, P. (2021, June). Application of explainable machine learning based on Catboost in credit scoring. *In Journal of Physics: Conference Series* (Vol. 1955, No. 1, p. 012039).
- Ruan, X., & Ding, N. (2021, July). Personal credit risk identification based on combined machine learning model. *In 2021 International Conference on Machine Learning and Intelligent Systems Engineering (MLISE)* (pp. 1-5).
- Rathore, A. S., Kumar, A., Tomar, D., Goyal, V., Sarda, K., & Vij, D. (2021, December). Credit card fraud detection using machine learning. *In 2021 10th International Conference on System Modeling & Advancement in Research Trends (SMART)* (pp. 167-171).
- Ribeiro, A. A. E., Lira, R. C., Macedo, M., Siqueira, H. V., & Bastos-Filho, C. (2023, September). Applying Reinforcement Learning for Multiple Functions in Swarm Intelligence. *In Brazilian Conference on Intelligent Systems* (pp. 197-212).

- Rodriguez-Perez, R., & Bajorath, J. (2019). Interpretation of compound activity predictions from complex machine learning models using local approximations and shapley values. *Journal of medicinal chemistry*, 63(16), 8761-8777.
- Raymaekers, J., Verbeke, W., & Verdonck, T. (2021). Weight-of-evidence 2.0 with shrinkage and spline-binning. *arXiv preprint arXiv: 2101.01494*.
- Szymura, A. (2022). Risk Assessment of Polish Joint Stock Companies: Prediction of Penalties or Compensation Payments. *Risks*, 10(5), 102.
- Stoltzfus, J. C. (2011). Logistic regression: a brief primer. *Academic emergency medicine*, 18(10), 1099-1104.
- Shi, S., Liu, D., & Zhao, Z. (2021, October). Parameter estimation of time fractional heat conduction equation for monocrystalline silicon. In *2021 China Automation Congress (CAC)* (pp. 3666-3671).
- Stratman, E., & Brandenburger, T. (2018). A New WOE and Scorecard Building R Package.
- Shannon, C. E. (1948). A mathematical theory of communication. *The Bell system technical journal*, 27(3), 379-423.
- Sharma, H., Andhalkar, A., Ajao, O., & Ogunleye, B. (2024). Analysing the Influence of Macroeconomic Factors on Credit Risk in the UK Banking Sector. *Analytics*, 3(1), 63-83.
- Sharma, V., Singh, A., Saxena, A. K., & Saxena, V. (2023, December). A Logistic Regression Based Credit Risk Assessment Using WoE Bining and Enhanced Feature Engineering Approach ANOVA and Chi-Square. In *2023 12th International Conference on System Modeling & Advancement in Research Trends (SMART)* (pp. 499-507).

- Shang, S. S., & Chiu, L. S. (2022). Leveraging smart technology for user experience personalization—A comparative case study of innovative payment systems. *Pacific Asia Journal of the Association for Information Systems*, 14(1), 6.
- Tiwari, P., Mehta, S., Sakhuja, N., Kumar, J., & Singh, A. K. (2021). Credit card fraud detection using machine learning: a study. *arXiv preprint arXiv: 2108.10005*.
- Tikhamarine, Y., Souag-Gamane, D., Ahmed, A. N., Kisi, O., & El-Shafie, A. (2020). Improving artificial intelligence models accuracy for monthly streamflow forecasting using grey Wolf optimization (GWO) algorithm. *Journal of Hydrology*, 582, 124435.
- Tang, P. L., Le Pham, T. D., & Dinh, T. B. (2022, September). Tree-Based Credit Card Fraud Detection Using Isolation Forest, Spectral Residual, and Knowledge Graph. *In International Conference on Machine Learning, Optimization, and Data Science* (pp. 326-340).
- Toh, Y. L. (2023). Addressing Traditional Credit Scores as a Barrier to Accessing Affordable Credit. *Economic Review*, 108(3), 12.
- Thamaraiselvi, R., Dhanasekaran, M., & Suresh, N. S. (2024). Advancements in solar photovoltaic modelling: selective opposition-based variable weighted grey wolf optimizer with improved Newton–Raphson analysis. *Electrical Engineering*, 1-32.
- VenkateswaraRao, M., Vellela, S., Reddy, V., Vullam, N., Sk, K. B., & Roja, D. (2023, March). Credit Investigation and Comprehensive Risk Management System based Big Data Analytics in Commercial Banking. *In 2023 9th International Conference on Advanced Computing and Communication Systems (ICACCS)* (Vol. 1, pp. 2387-2391).

- Wu, Y., Xu, Y., & Li, J. (2019). Feature construction for fraudulent credit card cash-out detection. *Decision Support Systems*, 127, 113155.
- Wang, S., Yan, X., Zheng, B., Wang, H., Xu, W., Peng, N., & Wu, Q. (2021, November). Risk and return prediction for pricing portfolios of non-performing consumer credit. *In Proceedings of the Second ACM International Conference on AI in Finance* (pp. 1-9).
- Wu, Y., Zhang, Q., Hu, Y., Sun-Woo, K., Zhang, X., Zhu, H., & Li, S. (2022). Novel binary logistic regression model based on feature transformation of XGBoost for type 2 Diabetes Mellitus prediction in healthcare systems. *Future Generation Computer Systems*, 129, 1-12.
- Wu, G., Mallipeddi, R., & Suganthan, P. N. (2017). Problem definitions and evaluation criteria for the CEC 2017 competition on constrained real-parameter optimization. *In 2017 IEEE congress on evolutionary computation (CEC)* (pp. 2443-2450). *IEEE*.
- Xie, R., Liu, R., Liu, X. B., & Zhu, J. M. (2021). Evaluation of SMEs' credit decision based on support vector machine-logistics regression. *Journal of Mathematics*, 2021, 1-10.
- Xia, Y., Liu, C., Li, Y., & Liu, N. (2017). A boosted decision tree approach using Bayesian hyper-parameter optimization for credit scoring. *Expert systems with applications*, 78, 225-241.
- Yang, K., Yuan, H., & Lau, R. Y. (2022). PsyCredit: An interpretable deep learning-based credit assessment approach facilitated by psychometric natural language processing. *Expert Systems with Applications*, 198, 116847.
- Yang, J., & Luo, D. (2017). The P2P risk assessment model based on the improved AdaBoost-SVM algorithm. *Journal of Financial Risk Management*, 6(2), 201-209.

- Yazna Sai, K., Venkata Bhavana, R., & Sudha, N. (2023, May). Detection of Fraudulent Credit Card Transactions Using Advanced LightGBM Approach. *In Machine Intelligence Techniques for Data Analysis and Signal Processing: Proceedings of the 4th International Conference MISIP 2022, Volume 1* (pp. 93-103).
- Yu, L., Zhou, R., Chen, R., & Lai, K. K. (2022). Missing data preprocessing in credit classification: One-hot encoding or imputation. *Emerging Markets Finance and Trade*, 58(2), 472-482.
- Yao, X., Liu, Y., & Lin, G. (1999). Evolutionary programming made faster. *IEEE Transactions on Evolutionary computation*, 3(2), 82-102.
- Yu, C., Jin, Y., Xing, Q., Zhang, Y., Guo, S., & Meng, S. (2024). Advanced User Credit Risk Prediction Model using LightGBM, XGBoost and Tabnet with SMOTEENN. *arXiv preprint arXiv: 2408.03497*.
- Yao, X., Fu, X., & Zong, C. (2022). Short-term load forecasting method based on feature preference strategy and LightGBM-XGboost. *IEEE Access*, 10, 75257-75268.
- Zhang, M., & Pang, L. J. (2019, January). Review of domestic application research of big data mining technology-SVM in credit risk evaluation. *In 3rd International Seminar on Education Innovation and Economic Management (SEIEM 2018)* (pp. 247-250).
- Zhou, J., Li, W., Wang, J., Ding, S., & Xia, C. (2019). Default prediction in P2P lending from high-dimensional data based on machine learning. *Physica A: Statistical Mechanics and its Applications*, 534, 122370
- Zhao, Y. (2023, August). A Credit Card Default Prediction Method Based on CatBoost. *In 2023 2nd International Conference on Artificial Intelligence, Internet and Digital Economy (ICAID 2023)* (pp. 178-184).

## LIST OF APPENDICES

**Appendix A:** Below is a screenshot of the experimental dataset set used in this study

| #  | A       | B        | C       | D       | E      | F     | G      | H        | I   | J    | K        | L    | M   | N        | O    | P              | Q               | R       | S      | T | U |
|----|---------|----------|---------|---------|--------|-------|--------|----------|-----|------|----------|------|-----|----------|------|----------------|-----------------|---------|--------|---|---|
| 1  | Account | Duration | Payment | Purpose | Credit | Value | Length | Instalme | Sex | Guar | Duration | Rest | Age | (Concurr | Type | No. of Occupat | No. of Telephon | Foreign | target |   |   |
| 2  | 1       | 18       | 4       | 2       | 1049   | 1     | 2      | 4        | 2   | 1    | 4        | 2    | 21  | 3        | 1    | 1              | 3               | 1       | 1      | 1 | 0 |
| 3  | 1       | 9        | 4       | 0       | 2799   | 1     | 3      | 2        | 3   | 1    | 2        | 1    | 36  | 3        | 1    | 2              | 3               | 2       | 1      | 1 | 0 |
| 4  | 2       | 12       | 2       | 9       | 841    | 2     | 4      | 2        | 2   | 1    | 4        | 1    | 23  | 3        | 1    | 1              | 2               | 1       | 1      | 1 | 0 |
| 5  | 1       | 12       | 4       | 0       | 2132   | 1     | 3      | 3        | 3   | 1    | 2        | 1    | 39  | 3        | 1    | 2              | 2               | 2       | 1      | 2 | 0 |
| 6  | 1       | 12       | 4       | 0       | 2171   | 1     | 3      | 4        | 3   | 1    | 4        | 2    | 38  | 1        | 2    | 2              | 2               | 1       | 1      | 2 | 0 |
| 7  | 1       | 10       | 4       | 0       | 2241   | 1     | 2      | 1        | 3   | 1    | 3        | 1    | 48  | 3        | 1    | 2              | 2               | 2       | 1      | 2 | 0 |
| 8  | 1       | 8        | 4       | 0       | 3398   | 1     | 4      | 1        | 3   | 1    | 4        | 1    | 39  | 3        | 2    | 2              | 2               | 1       | 1      | 2 | 0 |
| 9  | 1       | 6        | 4       | 0       | 1361   | 1     | 2      | 2        | 3   | 1    | 4        | 1    | 40  | 3        | 2    | 1              | 2               | 2       | 1      | 2 | 0 |
| 10 | 4       | 18       | 4       | 3       | 1098   | 1     | 1      | 4        | 2   | 1    | 4        | 3    | 65  | 3        | 2    | 2              | 1               | 1       | 1      | 1 | 0 |
| 11 | 2       | 24       | 2       | 3       | 3758   | 3     | 1      | 1        | 2   | 1    | 4        | 4    | 23  | 3        | 1    | 1              | 1               | 1       | 1      | 1 | 0 |
| 12 | 1       | 11       | 4       | 0       | 3905   | 1     | 3      | 2        | 3   | 1    | 2        | 1    | 36  | 3        | 1    | 2              | 3               | 2       | 1      | 1 | 0 |
| 13 | 1       | 30       | 4       | 1       | 6187   | 2     | 4      | 1        | 4   | 1    | 4        | 3    | 24  | 3        | 1    | 2              | 3               | 1       | 1      | 1 | 0 |
| 14 | 1       | 6        | 4       | 3       | 1957   | 1     | 4      | 1        | 2   | 1    | 4        | 3    | 31  | 3        | 2    | 1              | 3               | 1       | 1      | 1 | 0 |
| 15 | 2       | 48       | 3       | 10      | 7582   | 2     | 1      | 2        | 3   | 1    | 4        | 4    | 31  | 3        | 2    | 1              | 4               | 1       | 2      | 1 | 0 |
| 16 | 1       | 18       | 2       | 3       | 1936   | 5     | 4      | 2        | 4   | 1    | 4        | 3    | 23  | 3        | 1    | 2              | 2               | 1       | 1      | 1 | 0 |
| 17 | 1       | 6        | 2       | 3       | 2647   | 3     | 3      | 2        | 3   | 1    | 3        | 1    | 44  | 3        | 1    | 1              | 3               | 2       | 1      | 1 | 0 |
| 18 | 1       | 11       | 4       | 0       | 3939   | 1     | 3      | 1        | 3   | 1    | 2        | 1    | 40  | 3        | 2    | 2              | 2               | 2       | 1      | 1 | 0 |
| 19 | 2       | 18       | 2       | 3       | 3213   | 3     | 2      | 1        | 4   | 1    | 3        | 1    | 25  | 3        | 1    | 1              | 3               | 1       | 1      | 1 | 0 |
| 20 | 2       | 36       | 4       | 3       | 2337   | 1     | 5      | 4        | 3   | 1    | 4        | 1    | 36  | 3        | 2    | 1              | 3               | 1       | 1      | 1 | 0 |
| 21 | 4       | 11       | 4       | 0       | 7228   | 1     | 3      | 1        | 3   | 1    | 4        | 2    | 39  | 3        | 2    | 2              | 2               | 1       | 1      | 1 | 0 |
| 22 | 1       | 6        | 4       | 0       | 3676   | 1     | 3      | 1        | 3   | 1    | 3        | 1    | 37  | 3        | 1    | 3              | 3               | 2       | 1      | 1 | 0 |
| 23 | 2       | 12       | 4       | 0       | 3124   | 1     | 2      | 1        | 3   | 1    | 3        | 1    | 49  | 1        | 2    | 2              | 2               | 2       | 1      | 1 | 0 |
| 24 | 2       | 36       | 2       | 5       | 2384   | 1     | 2      | 4        | 3   | 1    | 1        | 4    | 33  | 3        | 1    | 1              | 2               | 1       | 1      | 1 | 1 |
| 25 | 2       | 12       | 4       | 4       | 1424   | 1     | 4      | 4        | 3   | 1    | 3        | 2    | 26  | 3        | 2    | 1              | 3               | 1       | 1      | 1 | 0 |
| 26 | 1       | 6        | 4       | 0       | 4716   | 5     | 2      | 1        | 3   | 1    | 3        | 1    | 44  | 3        | 2    | 2              | 2               | 2       | 1      | 1 | 0 |
| 27 | 2       | 11       | 3       | 3       | 4771   | 1     | 4      | 2        | 3   | 1    | 4        | 2    | 51  | 3        | 2    | 1              | 3               | 1       | 1      | 1 | 0 |
| 28 | 1       | 12       | 2       | 2       | 652    | 1     | 5      | 4        | 2   | 1    | 4        | 2    | 24  | 3        | 1    | 1              | 3               | 1       | 1      | 1 | 0 |
| 29 | 2       | 9        | 4       | 3       | 1154   | 1     | 5      | 2        | 3   | 1    | 4        | 1    | 37  | 3        | 2    | 3              | 2               | 1       | 1      | 1 | 0 |
| 30 | 4       | 15       | 2       | 0       | 3556   | 5     | 3      | 3        | 3   | 1    | 2        | 4    | 29  | 3        | 2    | 1              | 3               | 1       | 1      | 1 | 0 |
| 31 | 3       | 42       | 4       | 1       | 4796   | 1     | 5      | 4        | 3   | 1    | 4        | 4    | 56  | 3        | 3    | 1              | 3               | 1       | 1      | 1 | 0 |
| 32 | 3       | 30       | 4       | 3       | 3017   | 1     | 5      | 4        | 3   | 1    | 4        | 2    | 47  | 3        | 2    | 1              | 3               | 1       | 1      | 1 | 0 |
| 33 | 4       | 36       | 4       | 0       | 3535   | 1     | 4      | 4        | 3   | 1    | 4        | 3    | 37  | 3        | 2    | 2              | 3               | 1       | 2      | 1 | 0 |
| 34 | 4       | 36       | 4       | 0       | 6614   | 1     | 5      | 4        | 3   | 1    | 4        | 3    | 34  | 3        | 2    | 2              | 4               | 1       | 2      | 1 | 0 |
| 35 | 4       | 24       | 2       | 3       | 1376   | 3     | 4      | 4        | 2   | 1    | 1        | 3    | 28  | 3        | 2    | 1              | 3               | 1       | 1      | 1 | 0 |
| 36 | 1       | 15       | 2       | 0       | 1721   | 1     | 2      | 2        | 3   | 1    | 3        | 1    | 34  | 3        | 2    | 1              | 3               | 1       | 1      | 1 | 0 |

**Appendix B:** The code used for model development is python language. The detailed module code screenshot is as follows

```
# -*- coding: utf-8 -*-
# step1_woe.py
import numpy as np
import pandas as pd
from sklearn.cluster import KMeans
from statsmodels.stats.outliers_influence import variance_inflation_factor
import statsmodels.api as sm
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from sklearn.externals import joblib
from sklearn.metrics import accuracy_score

def woe_more(item, df, df_woe):
    xitem = np.array(df[item])
    y = df.loc[:, 'target']
    y = np.array(y)
    x = []
    for k in xitem:
        x.append(k)
    leasentro = 100
    tt_bad = sum(y)
    tt_good = len(y) - sum(y)
    l = []
    #3
    for m in range(10):
        y_pred = KMeans(n_clusters=4, random_state=m).fit_predict(x)
        a = [[[], []], [[], []], [[], []], [[], []]]
        for i in range(len(y_pred)):
            a[y_pred[i]][0].append(x[i][0])
            a[y_pred[i]][1].append(y[i])
        a = sorted(a, key=lambda x: sum(x[0]) / len(x[0]))
        if sum(a[0][1]) / len(a[0][1]) >= sum(a[1][1]) / len(a[1][1]) >= sum(a[2][1]) / len(a[2][1]) >= sum(a[3][1]) \
            / len(a[3][1]) or sum(a[0][1]) / len(a[0][1]) <= sum(a[1][1]) / len(a[1][1]) \
            <= sum(a[2][1]) / len(a[2][1]) <= sum(a[3][1]) / len(a[3][1]):
            entro = 0
            for j in a:
                entro = entro + (- (len(j[1]) - sum(j[1])) / len(j[1]) * np.log((len(j[1]) - sum(j[1])) \
                    / len(j[1])) - sum(
                        j[1]) / len(j[1]) * np.log(sum(j[1]) / len(j[1]))
            if entro < leasentro:
                leasentro = entro
                l = []
                for k in a:
                    l.append([min(k[0]), max(k[0]), np.log((sum(k[1]) / (len(k[1]) - sum(k[1])))) / (tt_bad / tt_good)),
                        sum(k[1]) / len(k[1])])
                    # print (sum(k[1]),len(k[1]))
    #4
    for m in range(10):
        y_pred = KMeans(n_clusters=5, random_state=m).fit_predict(x)
        a = [[[], []], [[], []], [[], []], [[], []], [[], []]]
        for i in range(len(y_pred)):
            a[y_pred[i]][0].append(x[i][0])
            a[y_pred[i]][1].append(y[i])
        a = sorted(a, key=lambda x: sum(x[0]) / len(x[0]))
        if sum(a[0][1]) / len(a[0][1]) >= sum(a[1][1]) / len(a[1][1]) >= sum(a[2][1]) / len(a[2][1]) >= sum(a[3][1]) \
            / len(a[3][1]) >= sum(a[4][1]) / len(a[4][1]) or sum(a[0][1]) / len(a[0][1]) <= sum(a[1][1]) / len(
            a[1][1]) \
```

```

# -*- coding: utf-8 -*-
#step2_sklearn_model.py
import pickle
import math
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score
#import statsmodels.api as sm

from sklearn import metrics
from sklearn.metrics import roc_curve, auc, roc_auc_score
from sklearn.metrics import precision_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score

df_german=pd.read_excel("german_woe.xlsx")
#df_german=pd.read_excel("german_credit.xlsx")
#df_german=pd.read_excel("df_after_vif.xlsx")
y=df_german["target"]
#x=df_german.ix[:, "Account Balance": "Foreign Worker"]
x=df_german.ix[:, "Account Balance": "Foreign Worker"]
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=0)

#solver='liblinear'
classifier = LogisticRegression(solver='liblinear')
classifier.fit(X_train, y_train)
predictions = classifier.predict(X_test)

print("accuracy on the training subset: {:.3f}".format(classifier.score(X_train, y_train)))
print("accuracy on the test subset: {:.3f}".format(classifier.score(X_test, y_test)))
'''
P0 = 50
PDO = 10
theta0 = 1.0/20
B = PDO/np.log(2)
A = P0 + B*np.log(theta0)
def Score(probability):
    #e
    score = A-B*np.log(probability/(1-probability))
    return score
def List_score(pos_probability_list):
    list_score=[]
    for probability in pos_probability_list:
        score=Score(probability)
        list_score.append(score)
    return list_score

P0 = 50
PDO = 10

# -*- coding: utf-8 -*-
# step4_validation.py
import pickle
import math
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score
#import statsmodels.api as sm
from sklearn import metrics
from sklearn.metrics import roc_curve, auc, roc_auc_score
from sklearn.metrics import precision_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score

classifier_f = open("scoreCard.pickle", "rb")
classifier = pickle.load(classifier_f)

df_german=pd.read_excel("german_woe.xlsx")
y=df_german["target"]
x=df_german.ix[:, "Account Balance": "Foreign Worker"]
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=0)

y_true=y_test
y_pred=classifier.predict(X_test)
accuracyScore = accuracy_score(y_true, y_pred)
print(' model accuracy is:', accuracyScore)

#precision, TP/(TP+FP)
precision=precision_score(y_true, y_pred)
print(' model precision is:', precision)

#recall (sensitive) , (TP) / (TP+FN)
sensitivity=recall_score(y_true, y_pred)
print(' model sensitivity is:', sensitivity)

f1Score=f1_score(y_true, y_pred)
print(' f1_score:', f1Score)

probability_list=classifier.predict_proba(X_test)
pos_probability_list=[i[1] for i in probability_list]

def AUC(y_true, y_scores):
    auc_value=0
    #auc
    fpr, tpr, thresholds = metrics.roc_curve(y_true, y_scores, pos_label=1)
    auc_value= auc(fpr, tpr) #
    #print("fpr:", fpr)
    #print("tpr:", tpr)
    #print("thresholds:", thresholds)
    if auc_value<0.5:
        auc_value=1-auc_value

```

```

#Calculate the value of variable iv
def Get_IV(readFileName, saveFileName, score, target):
    df=pd.read_excel(readFileName)
    #Each score section counts
    total = df.groupby([score])[target].count()
    bad = df.groupby([score])[target].sum()
    all = pd.DataFrame({'total':total, 'bad':bad})
    #fraction
    num=total.sum()
    all['range_rate']=total/num
    #Good customer score proportion
    all['good'] = all['total'] - all['bad']
    #boxer is a boxer-divided device
    all['boxer'] = all.index
    all = all.sort_values(by="boxer", ascending=True)
    all.index = range(len(all))
    #Total bad customers
    num_bad=all['bad'].sum()
    #print("num_bad:", num_bad)
    #Number of good customers
    num_good= all['good'].sum()
    #print("num_good:", num_good)
    #Total sample size
    num_total=num_bad+num_good
    #print("num_total:", num_total)
    #The proportion of bad customers
    bad_per=all['bad']/ num_bad
    good_per=all['good']/ num_good

    all['bad_per']=bad_per
    all['good_per']=good_per
    p_bad_devide_p_good=bad_per/good_per
    all['p_bad_devide_p_good']=p_bad_devide_p_good
    #woe formula
    woe_formula=lambda x:math.log(x, math.e)
    woe=list(map(woe_formula, all['p_bad_devide_p_good']))
    all['woe']=woe
    p_bad_minus_p_good=all['bad_per']-all['good_per']

```

**Appendix C:** The core code of EBGWO-CatBoost algorithm is as follows

```

# The initial values of all parameters of the CatBoost algorithm
catboost_params = {
    'loss_function': 'RMSE',
    'custom_metric': 'None',
    'eval_metric': 'True',
    'random_strength': 1,
    'use_best_model': 'False',
    'ignored_features': 'None',
    'has_time': 'False',
    'nan_mode': 'Min',
    'fold_len_multiplier': 2,
    'approx_on_full_history': 'False',
    'class_weights': 'None',
    'allow_const_label': 'False',
    'iterations': 100,
    'learning_rate': 0.03,
    'l2_leaf_reg': 3,
    'depth': 6,
    'bagging_temperature': 1,
    'random_strength': 1,
    'rsm': 1,
    'one_hot_max_size': 2,
    'leaf_estimation_method': 'Gradient',
    'fold_len_multiplier': 2,
    'border_count': 128,
    # .....

```

```

# Function: CatBoost algorithm training data
def CatBoost_score(ind_var):
    x1 = ind_var[0]
    x2 = ind_var[1]
    CatBoost = CatBoostRegressor(**catboost_params) # Initialize all parameters of the CatBoost model
    # cv_scores = cross_val_score(CatBoost, x_train, y_train, cv=3, scoring='accuracy')
    # print(cv_scores.mean())
    CatBoost.fit(x_train, y_train)
    predict = CatBoost.predict(x_test)
    print('scores: ', r2_score(y_test, predict))
    return 1-accuracy_score(y_test, predict)

def EBGWO_CatBoost(CatBoost_score, lb, ub, dim, SearchAgents_no, Max_iter):
    # 初始化 alpha, beta, delta_pos and r
    Alpha_pos = np.zeros(dim) # Location. Forms a list of 2
    # means plus or minus infinity, numbers are less than +inf, Positive infinity:float("inf"); Negative infinity:float("-inf")
    Alpha_score = float("inf")
    Beta_pos = np.zeros(dim)
    Beta_score = float("inf")
    Delta_pos = np.zeros(dim)
    Delta_score = float("inf") # float() Function to convert integers and strings to floating-point numbers.
    r_pos = np.zeros(dim)
    r_score = float("inf")

    # Initialize all Wolf positions
    Positions = np.zeros((SearchAgents_no, dim))
    for i in range(dim):
        Positions[:, i] = np.ones(SearchAgents_no)*ub[i]
    Convergence_curve = np.zeros(Max_iter)

    # Iterate to find the optimal value
    for i in range(0, Max_iter):
        for j in range(0, SearchAgents_no):
            # Returns a search agent that exceeds the boundaries of the search space
            for j in range(dim): # 30
                Positions[i, j] = np.clip(Positions[i, j], lb[j], ub[j])
            # Compute the objective function for each search agent
            fitness = CatBoost_score(Positions[i, :]) # Bring a row of data into a function calculation

            if fitness < Alpha_score:
                Alpha_score = fitness
                Alpha_pos = Positions[i, :].copy()
            if (fitness > Alpha_score and fitness < Beta_score):
                Beta_score = fitness
                Beta_pos = Positions[i, :].copy()
            if (fitness > Alpha_score and fitness > Beta_score and fitness < Delta_score):
                Delta_score = fitness
                Delta_pos = Positions[i, :].copy()

            if (fitness > Alpha_score and fitness > Beta_score and fitness > Delta_score and fitness < r_score):
                r_score = fitness
                r_pos = Positions[i, :].copy()
        print(1-Alpha_score)
        a = 2 - 1 * ((2) / Max_iter) # a decreases linearly from 2 to 0
        for i in range(0, SearchAgents_no):
            for j in range(0, dim):
                r1 = random.random() # r1 is a random number in [0,1]
                r2 = random.random() # r2 is a random number in [0,1]
                A1 = 2 * a * r1 - a # Equation (3.3)
                C1 = 2 * r2 # Equation (3.4)
                # D_alpha indicates the distance between the candidate and the Alpha Wolf
                # lpha_pos[j] is the Alpha position, and Positions[i, j] is the position of the candidate gray Wolf
                D_alpha = abs(C1 * Alpha_pos[j] - Positions[i, j])
                # lpha_pos[j] is the Alpha position, and Positions[i, j] is the position of the candidate gray Wolf
                X1 = Alpha_pos[j] - A1 * D_alpha

                r1 = random.random()
                r2 = random.random()
                A2 = 2 * a * r1 - a
                C2 = 2 * r2
                D_beta = abs(C2 * Beta_pos[j] - Positions[i, j])
                X2 = Beta_pos[j] - A2 * D_beta

                r1 = random.random()
                r2 = random.random()
                A3 = 2 * a * r1 - a
                C3 = 2 * r2
                D_delta = abs(C3 * Delta_pos[j] - Positions[i, j])
                X3 = Delta_pos[j] - A3 * D_delta

                r1 = random.random()
                r2 = random.random()
                A4 = 2 * a * r1 - a
                C4 = 2 * r2
                D_r = abs(C4 * r_pos[j] - Positions[i, j])
                X4 = r_pos[j] - A4 * D_delta

            # Improved mathematical formula of EBGWO algorithm
            if (|A|<1):
                Positions[i, j] = (X1 + X2) / 2
            else:
                Positions[i, j] = (X1 + X2 + X3 + X4) / 4
        Convergence_curve[i] = Alpha_score

```

Appendix D: Other screenshots of the experiment run are as follows

| A   |                             |  | B                                 |  |  | C            |  |  |
|-----|-----------------------------|--|-----------------------------------|--|--|--------------|--|--|
| 87  |                             |  | No of dependents                  |  |  | 0            |  |  |
| 88  |                             |  | Telephone                         |  |  | 0            |  |  |
| 89  |                             |  | Foreign Worker                    |  |  | 0            |  |  |
| 90  |                             |  | target                            |  |  | 0            |  |  |
| 91  | Guarantors                  |  | Account Balance                   |  |  | -0.120906612 |  |  |
| 92  |                             |  | Duration of Credit (month)        |  |  | -0.012838099 |  |  |
| 93  |                             |  | Payment Status of Previous Credit |  |  | -0.046787212 |  |  |
| 94  |                             |  | Purpose                           |  |  | 0.009615575  |  |  |
| 95  |                             |  | Credit Amount                     |  |  | 0.000755834  |  |  |
| 96  |                             |  | Value Savings/Stocks              |  |  | -0.107099796 |  |  |
| 97  |                             |  | Length of current employment      |  |  | -0.017199245 |  |  |
| 98  |                             |  | Instalment per cent               |  |  | -0.014301446 |  |  |
| 99  |                             |  | Sex & Marital Status              |  |  | 0.043115856  |  |  |
| 100 |                             |  | Guarantors                        |  |  | 0            |  |  |
| 101 | Duration in Current address |  | Duration in Current address       |  |  | 0            |  |  |
| 102 |                             |  | Most valuable available asset     |  |  | 0            |  |  |
| 103 |                             |  | Age (years)                       |  |  | 0            |  |  |
| 104 |                             |  | Concurrent Credits                |  |  | 0            |  |  |
| 105 |                             |  | Type of apartment                 |  |  | 0            |  |  |
| 106 |                             |  | No of Credits at this Bank        |  |  | 0            |  |  |
| 107 |                             |  | Occupation                        |  |  | 0            |  |  |
| 108 |                             |  | No of dependents                  |  |  | 0            |  |  |
| 109 |                             |  | Telephone                         |  |  | 0            |  |  |
| 110 |                             |  | Foreign Worker                    |  |  | 0            |  |  |
| 111 | Duration in Current address |  | target                            |  |  | 0            |  |  |
| 112 |                             |  | Account Balance                   |  |  | -0.048577133 |  |  |
| 113 |                             |  | Duration of Credit (month)        |  |  | 0.043670502  |  |  |
| 114 |                             |  | Payment Status of Previous Credit |  |  | 0.059582222  |  |  |
| 115 |                             |  | Purpose                           |  |  | -0.056585596 |  |  |
| 116 |                             |  | Credit Amount                     |  |  | 0.023634399  |  |  |
| 117 |                             |  | Value Savings/Stocks              |  |  | 0.093086335  |  |  |
| 118 |                             |  | Length of current employment      |  |  | 0.256015756  |  |  |
| 119 |                             |  | Instalment per cent               |  |  | 0.047507168  |  |  |
| 120 |                             |  | Sex & Marital Status              |  |  | -0.033859199 |  |  |
| 121 | Duration in Current address |  | Guarantors                        |  |  | -0.023433522 |  |  |
| 122 |                             |  | Duration in Current address       |  |  | 0            |  |  |

| A  |  |  | B                                 |  |  | C           |  |  |
|----|--|--|-----------------------------------|--|--|-------------|--|--|
| 0  |  |  | Telephone                         |  |  | 0.006377605 |  |  |
| 1  |  |  | Occupation                        |  |  | 0.00809505  |  |  |
| 2  |  |  | Age (years)                       |  |  | 0.010806344 |  |  |
| 3  |  |  | No of Credits at this Bank        |  |  | 0.01142553  |  |  |
| 4  |  |  | Guarantors                        |  |  | 0.01642028  |  |  |
| 5  |  |  | Purpose                           |  |  | 0.023892285 |  |  |
| 6  |  |  | Instalment per cent               |  |  | 0.025568865 |  |  |
| 7  |  |  | Type of apartment                 |  |  | 0.038878144 |  |  |
| 8  |  |  | Sex & Marital Status              |  |  | 0.042633391 |  |  |
| 9  |  |  | Foreign Worker                    |  |  | 0.043877412 |  |  |
| 10 |  |  | Concurrent Credits                |  |  | 0.057614542 |  |  |
| 11 |  |  | Most valuable available asset     |  |  | 0.07831382  |  |  |
| 12 |  |  | Length of current employment      |  |  | 0.082864571 |  |  |
| 13 |  |  | Credit Amount                     |  |  | 0.0952721   |  |  |
| 14 |  |  | Value Savings/Stocks              |  |  | 0.165819696 |  |  |
| 15 |  |  | Duration of Credit (month)        |  |  | 0.22510035  |  |  |
| 16 |  |  | Payment Status of Previous Credit |  |  | 0.291829855 |  |  |
| 17 |  |  | Account Balance                   |  |  | 0.634596774 |  |  |

Docer-在线模板

工作簿1

20-mutual.xls

110-mutual.xls\*

格式转换

<

```

EBGWO-code
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

def ebgwo(objective_function, bounds, population_size, iterations):
    dimension = len(bounds)
    alpha_position = np.zeros(dimension)
    alpha_score = float("inf")
    beta_position = np.zeros(dimension)
    beta_score = float("inf")
    delta_position = np.zeros(dimension)
    delta_score = float("inf")
    r_position = np.zeros(dimension)
    r_score = float("inf")
    positions = np.random.rand(population_size, dimension) * (bounds[:, 1] - bounds[:, 0]) + bounds[:, 0]
    for t in range(iterations):
        for i in range(population_size):
            fitness = objective_function(positions[i])
            if fitness < alpha_score:
                alpha_score = fitness
                alpha_position = positions[i].copy()
            if fitness > alpha_score and fitness < beta_score:
                beta_score = fitness
                beta_position = positions[i].copy()
            if fitness > alpha_score and fitness > beta_score and fitness < delta_score:
                delta_score = fitness
                delta_position = positions[i].copy()
            if fitness > alpha_score and fitness > beta_score and fitness < delta_score:
                delta_score = fitness
                delta_position = positions[i].copy()
            if (fitness > alpha_score and fitness > beta_score and fitness > delta_score and fitness < r_score):
                r_score = fitness
                r_position = positions[i].copy()
        a = 2 - 2 * t / iterations
        for i in range(population_size):
            for j in range(dimension):
                r1 = np.random.random()
                r2 = np.random.random()
                A1 = 2 * a * r1 - a
                C1 = 2 * r2
                D_alpha = abs(C1 * alpha_position[j] - positions[i, j])
                X1 = alpha_position[j] - A1 * D_alpha
                r1 = np.random.random()
                r2 = np.random.random()
                A2 = 2 * a * r1 - a
                C2 = 2 * r2
                D_beta = abs(C2 * beta_position[j] - positions[i, j])
                X2 = beta_position[j] - A2 * D_beta
                r1 = np.random.random()
                r2 = np.random.random()
                A3 = 2 * a * r1 - a
                C3 = 2 * r2
                D_delta = abs(C3 * delta_position[j] - positions[i, j])
                X3 = delta_position[j] - A3 * D_delta
                r1 = np.random.random()
                r2 = np.random.random()
                A4 = 2 * a * r1 - a
                C4 = 2 * r2
                D_r = abs(C4 * r_position[j] - positions[i, j])
                X4 = r_position[j] - A4 * D_r
                if (abs(a) < 1):
                    #X = (X1 + X2) / 2
                else:
                    #X = (X1 + X2 + X3 + X4) / 4
    return alpha_position, alpha_score

```