Risk-Sensitive Credit Strategy Optimization With Reinforcement Learning

Group 2:

CHANG WEN-YU (01505913)

HONG YANG (01502068)

LI SINUAN (01503010)

PHOO PYAE HSU MYAT (01520414)

ZHANG YINLIANG (01345913)

01

Introduction & Feature Definition

Why This Matters

- Traditional methods rely on **fixed rules** or **income-based policies**
- Fail to adapt to **customer behavior changes**
- Result in **inefficient capital allocation**: Too much risk from **high-risk customers,** Missed revenue from **low-risk customers**

Objective

Develop a **data-driven framework** for **optimizing credit limit decisions**.

- Identify which customers should receive **limit increases**
- Improve **long-term profitability**
- Maintain **acceptable credit risk exposure**

Our Approach

By modeling **repayment patterns** and simulating **limit changes**, we aim to:

- Improve decision quality
- Increase interest revenue
- Reduce losses from defaults

Methodological Overview:

Data Preparation & Feature Engineering	Clean and transform historical credit card usage data into meaningful features for modeling.
Simulator Construction (Classification + Regression)	Build a two-stage supervised ML model to simulate customer behavior.
RL Agent Training	Use DQN in a simulated offline environment.
Policy Evaluation & Backtesting	Compare learned policy against baselines on financial outcomes.
Deployment & Interpretation	Visualize policy decisions and explain feature contributions using SHAP.

Data Source

The dataset is sourced from the ***Home Credit Default Risk*** competition on **Kaggle**, which provides rich, anonymized credit information at the customer level.

File Name	Description
application_train.csv	Core customer profile and loan application data
credit_card_balance.csv	Monthly credit card usage and payment information
installments_payments.csv	Payment records for past installment loans
POS_CASH_balance.csv	Point-of-sale and cash loan history
bureau.csv + bureau_balance.csv	Third-party credit bureau records and status tracking
previous_application.csv	Records of all prior loan applications and outcomes

Data Wrangling

Features

1	MONTHS_BALANCE(-1 means the freshest balance date)	`-3, -2, -1` as the future data window and `-6,-5,-4` as the prediction window.
2	UR: Utilization Rate (avg over 3 months)	$\mathrm{UR} = \frac{1}{3}\sum_{i=1}^{3} \frac{\text{Outstanding Balance}_i}{\text{Credit Limit}_i}$
3	PR: Payment Rate (avg over 3 months)	$\mathrm{PR} = \frac{1}{3}\sum_{i=1}^{3} \frac{\text{AMT\_PAYMENT}_i}{\text{AMT\_TOTAL\_RECEIVABLE}_i}$
4	TC_i:Total consumer spending in month i (i = 1, 2, 3)	TC_1 = spending in the most recent (last) month
5	EO_i:Operative state in month i: consecutive missed payments	$\mathrm{EO}_1 = \text{non\_pay}_1 \mid \mathrm{EO}_2 = \text{non\_pay}_2 \cdot (\mathrm{EO}_1 + 1) \mid \mathrm{EO}_3 = \text{non\_pay}_3 \cdot (\mathrm{EO}_2 + 1)$
6	MP_R:Total number of missed payments in retrospective window	$\mathrm{MP\_R} = \sum_{i=1}^{3} 1\left(\text{SK\_DPD\_DEF}_i > 0\right)$

Data Wrangling

Features

7	OB_cday_i: Outstanding balance in month i	$\text{OB\_cday}_i = \text{AMT\_BALANCE at MONTHS\_BALANCE} = -i \quad \text{for } i = 1, 2, 3$
8	P_pday_i: Payment ratio at payment date	$P\_pday_i = \frac{\text{AMT\_PAYMENT\_TOTAL\_CURRENT}_i}{\text{AMT\_BALANCE}_i}$
9	BS: Bureau score proxy	$\text{BS} = \frac{1}{N} \sum_{j=1}^{N} \text{CREDIT\_DAY\_OVERDUE}_j$
10	EI: Estimated Income	$\text{EI} = \text{AMT\_INCOME\_TOTAL}$
11	INT: Interest Rate	Fill NaN with fallback value $\rightarrow$ 0.1887
12	N_Months_R:Active months since account opened	$\text{N\_Months\_R} = \text{Count of unique MONTHS\_BALANCE entries per customer}$

Data Wrangling

Synthetic Features

13 L_R: Credit Limit in Retrospective Window

14 L_P: Prospective Credit Limit

15 HA_P: A binary flag indicating if L_P > L_R

Calculation:

$$L_P = \begin{cases} L_R \times \beta & \text{if customer receives an increase} \\ L_R & \text{otherwise} \end{cases}$$

- **β**: Fixed multiplier (1.5)
- **HA_P**: A binary flag indicating if `L_P > L_R`

Implementation Notes:

- A random sample (20%) of customers can be assigned a limit increase
- Alternatively, the decision can be rule-based (only customers with high payment rate and no missed payments)

02

Classify Balance Class Under Supervision Model

Setting Balance Class Threshold

Why Segment by Balance?

Captures Behavioral Structure

- Distinguishes fundamentally different customer types based on balance behavior
- Enables tailored modeling for groups like full payers, light users, and heavy revolvers

Supports Two-Stage Modeling Design

- Classifier pre-filters customers, identifying those likely to carry a balance
- Allows regressors to focus on relevant cases, avoiding noise from zero-balance customers

Benefits

Sharper Provision Estimates

- Focuses credit risk calculations on accounts with actual exposure
- Leads to more accurate and reliable provisioning

Avoids Unnecessary Credit Exposure

- Prevents limit increases for inactive or low-usage customers
- Helps reduce potential losses from unproductive credit

Strengthens RL Policy Outcomes

- Cleaner balance predictions improve reward signal quality
- Helps the RL agent learn safer and more profitable policies

Setting Balance Class Threshold

Distribution Based on AVG_BALANCE

(3-month before decision point average)

Medium=0	50%	Half of users carry no balance
75th≈ 81,397	75%	Top 25% have balances > 81K
90th≈ 233,371	90%	Top 10% carry > 230k

Choose **75K** as a threshold, defining top 25% as high usage customers

	Class 2	Class 0	Class 1
Threshold	AVG_BALANCE = 0	0 < AVG_BALANCE ≤ 75,000	AVG_BALANCE > 75,000
Description	**No balance:** Full payers or inactive accounts	**Low to moderate balance:** Moderate usage, Low Risk	**High balance:** High usage, Higher risk

Model Training Overview

Use historical customer behavior to **predict future balance behavior class**, enabling smarter credit policy decisions.

Inputs

Past 3-month behavioral indicators (MONTHS_BALANCE -7, -8, -9)

Labels

Class 0,1,2 based on 3-month current average balance (MONTHS_BALANCE -4,-5,-6)

Training

Models Tested

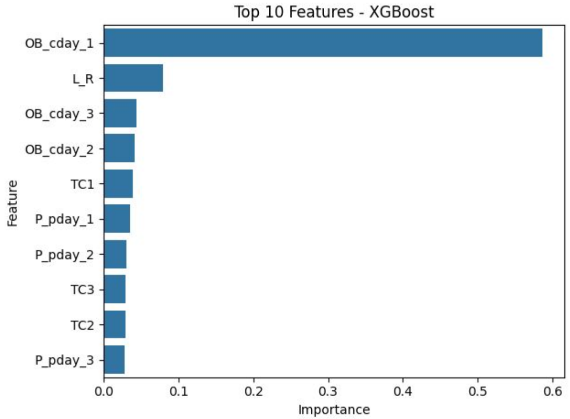
XGBoost,
LightGBM,
Random Forest,
Logistic Regression

Predict

Performance:
Accuracy: 88.43%
Macro F1-score: 85.80%

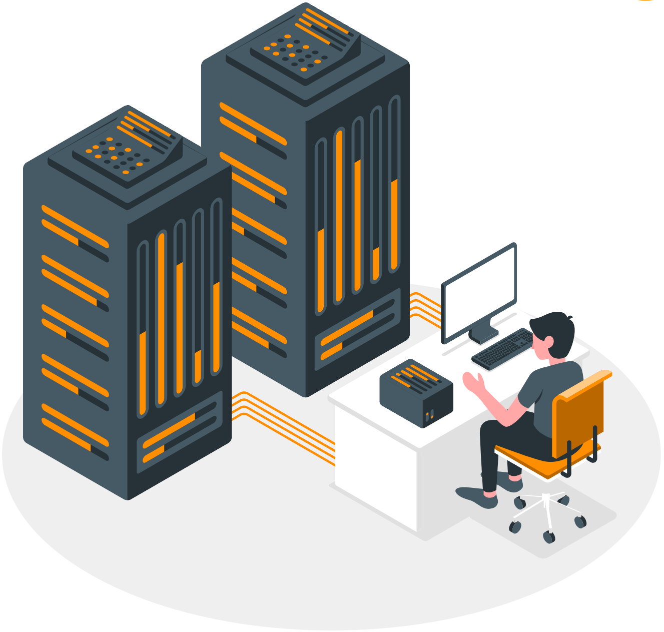
```
=== XGBoost ===
              precision    recall  f1-score   support

           0       0.80      0.79      0.80      2963
           1       0.92      0.92      0.92      7354

    accuracy                           0.88     10317
   macro avg       0.86      0.86      0.86     10317
weighted avg       0.88      0.88      0.88     10317
```

```
 Model Ranking (on Validation Set):
                 Model  Accuracy  Macro F1-score
2              XGBoost  0.884366        0.858081
3             LightGBM  0.884366        0.857642
1        Random Forest  0.882039        0.854869
0  Logistic Regression  0.862557        0.833841
```



03

Building the RL Environment for Credit Decisions



Transforming Supervised Outputs into RL State:

Preprocessing financial behaviors into state representations for RL training

State Feature Construction:

1. Removed redundant features
2. Merged 3-month rolling average of UR and PR
3. Filled missing values with median
4. Binned ΔPROVISION for discrete RL state

Final State Representation:

- **Utilization Rate**
- **ΔPROVISION_bin**
- **Payment Rate**
- **BALANCE_CLASS = 0,1 (From Phase 1 model)**

.	PAYMENT_RATE	UTIL_RATE	L_P	HA_P	UR	PR	UR_bin	PR_bin	ΔPROVISION	ΔPROVISION_bin
.	0.579943	0.061619	225000.0	0	0.061619	0.579943	1	11	0.0	49
.	0.662500	0.372428	225000.0	0	0.372428	0.662500	7	13	0.0	49
.	0.066436	0.418133	45000.0	0	0.418133	0.066436	8	1	0.0	49
.	0.106371	0.870598	135000.0	0	0.870598	0.106371	17	2	0.0	49
.	0.032796	0.240578	810000.0	0	0.240578	0.032796	4	0	0.0	49
.	...	...	...	...	...	...	...	...	...	...
.	0.035868	0.501507	135000.0	0	0.501507	0.035868	10	0	0.0	49
.	0.034624	0.644964	180000.0	0	0.644964	0.034624	12	0	0.0	49
.	0.052997	0.190506	180000.0	0	0.190506	0.052997	3	1	0.0	49
.	0.104444	0.634467	225000.0	0	0.634467	0.104444	12	2	0.0	49
.	0.005809	1.042110	175500.0	0	1.042110	0.005809	0	0	0.0	49

Simulated Environment for Credit Policy Learning :

- Agent observes customer
- Takes action → environment simulates outcome
- Reward from income vs risk
- State updates → next decision

Credit Agent

State

- Utilization Rate
- Payment Rate
- Balance Class
- ΔPROVISION_bin

Reward

(Interest Gain-Provision cost)

Action

- **0 = Maintain current credit limit**
- **1 = Increase credit limit by 20%**

Environment

- Simulates customer outcome using regressors to predict next 3-month balance
- Updates ΔPROVISION_bin in new state

Reward Function Design: Balancing Profit and Risk

Agent receives a risk-adjusted interest gain - expected loss

Reward Function =

Interest Income	−	Expected Provision Loss
$3 \times INT \times BAL \times (1 - PD)$		$PD \times LGD \times (BAL\_3 + CCF \times (L\_P - BAL\_3))$

- INT: Customer's interest rate
- BAL: Current balance
- PD: Probability of default
  (Assigned by BALANCE_CLASS)

- BAL_3: Predicted 3-month balance
- L_P: New credit limit after action
- LGD: Loss Given Default (50%)
- CCF: Credit Conversion Factor (80%)
- L_P – BAL_3: Unused Credit Limit

Reward guides the agent to balance revenue growth with risk control in credit policy decisions.

Simulated Environment Example:

Credit Agent

1

State

- Utilization Rate 61%
- Payment Rate 3%
- **Balance Class = 1 (High Risk)**
- ΔPROVISION_bin 49

2

Action

- **Action = 0** (Keep credit limit unchanged)

Environment

3

- Predicts next 3-month balance using regressors
- Updates state and feeds back to agent

4

Reward

(Interest Gain-Provision cost)= $376

04

Reinforcement Learning - Double Q Network

Why Use DQN for Credit Policy Optimization?

- Handles continuous state space (utilization rate, payment rate, etc.).
- Learns optimal actions via trial and error.
- Balances long-term rewards, not just immediate gains.

Q-Table (Q-learning)

- Typically discrete
- Explicit table Q(s, a)
- Uses Bellman equation to update table cells
- Simple, rule-based systems

DQN (Deep Q-Network)

- Can handle continuous or high-dimensional inputs
- Neural network approximates Q(s, a)
- Minimizes TD error (MSE) through neural networks
- Large-scale real-world problems

DQN Model Architecture

Network Structure:

- Input: state_dim = 4
- Hidden layers: 2 × Linear(64), ReLU
- Output: Q-values for 2 actions

```
def __init__(self, input_dim, output_dim):
    super(DQN, self).__init__()
    self.fc1 = nn.Linear(input_dim, 64)
    self.fc2 = nn.Linear(64, 64)
    self.out = nn.Linear(64, output_dim)
```

➡ **Lightweight and fast for convergence**

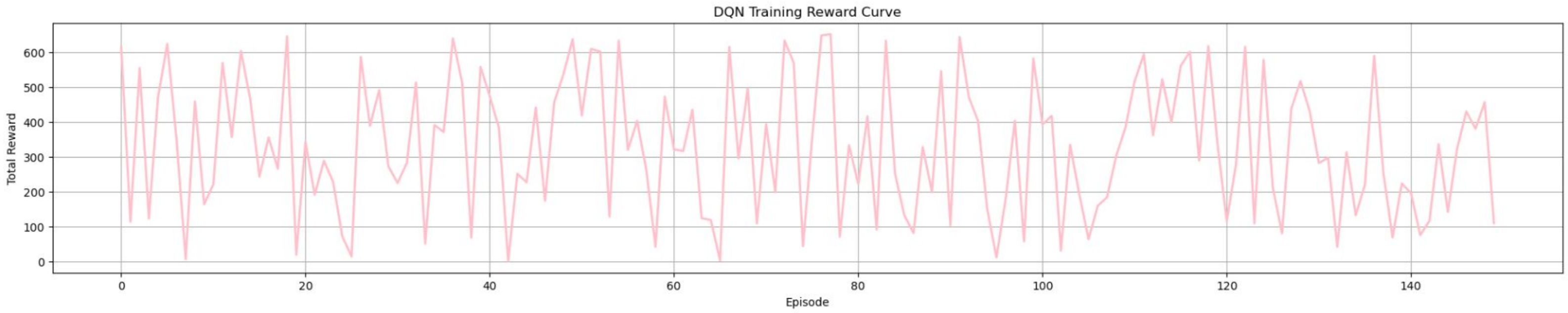
➡ **Sufficiently expressive for a small discrete action space**

DQN Key Hyperparameters

Parameter	Value	Purpose
Episodes	100	Training cycles
γ (Gamma)	0.9	Discount factor for future rewards
ε-decay	0.995 → 0.05	Epsilon-greedy exploration
Batch size	64	Sample size for replay buffer
Target Update	every 10 episodes	Stabilize Q-learning
Optimizer	Adam	Learning Q-network

Training Result: Reward Curve

- High volatility across 200-episode
- Upper reward envelope increases gradually
- High-reward episodes become more frequent over time
- Indicates partial learning progress and agent’s ability to exploit favorable cases



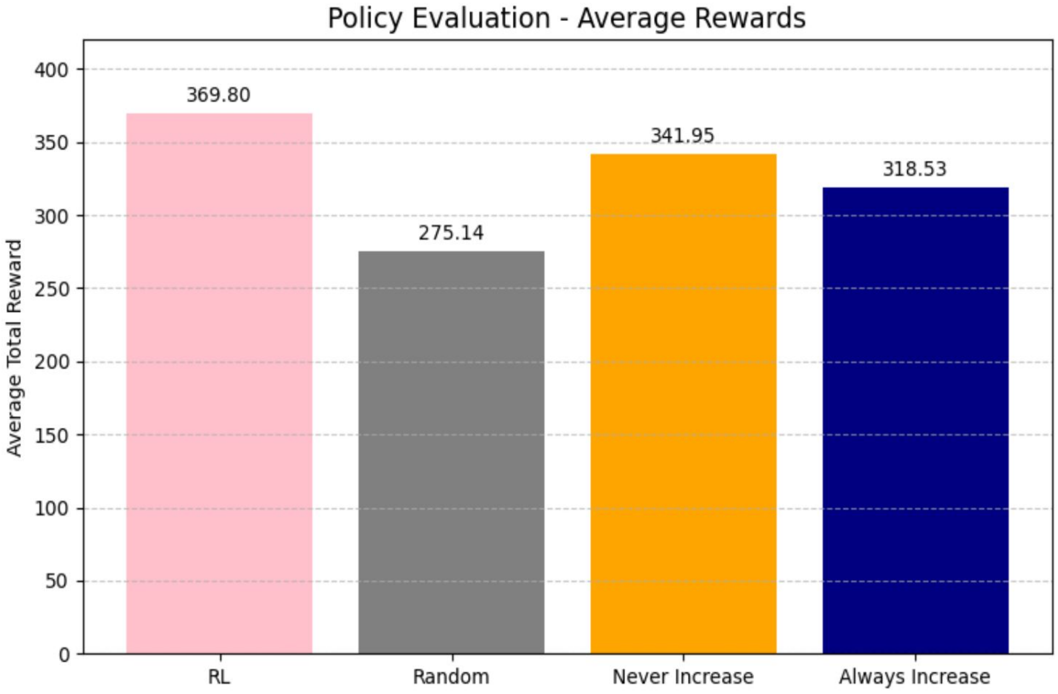
Reward Curve Instability Analysis

Possible Causes:	Possible Solutions:
Environment stochasticity - Customer sequence and response vary **Volatile reward design -** Rewards depend on predicted balances, INT, PD **Cold Start Instability -** Early Q-network is untrained, transitions still limited	Shuffle customer order each episode Train over more episodes to average out randomness Clip or normalize rewards to reduce extreme value swings Modularize reward components (e.g., separate interest and provision) Warm up replay buffer with 2–3× batch size before training starts Use running/moving average to smooth reward tracking

Action Distribution Analysis

- **Action 0:** 1,040,609 times (≈ 55.3%) (Keep credit limit unchanged)
- **Action 1**: 840,969 times (≈ 44.7%)
- The agent shows a slight preference for Action 0
- Suggests Action 0 may be slightly more rewarding or safer

Policy Evaluation



Benchmark Comparison

- Maintain Limit
- Always Increase
- Random

DQN RL Strategy

- Achieved **highest average reward**
- **Significantly outperformed** all baseline strategies
- Demonstrated ability to learn an **effective dynamic credit limit policy**

05

Conclusion and Future Directions

Conclusion & Future Directions

Key Takeaways

- Successfully implemented a functioning **DQN-based credit limit optimizer**.
- Balanced **revenue and credit risk** using simulated customer behavior and financial outcomes.
- Demonstrated proof of concept for **data-driven decision support** in consumer credit portfolios.

Major Future Improvements

- **Personalized Default Prediction**: Replace fixed PD with customer-level probability of default models.
- **Markov-Based Transitions**: Model behavior evolution over time using **state transition dynamics**.
- **Multi-Period Optimization**: Track cumulative rewards to reflect long-term profitability.
- **Feature Enrichment**: Integrate **repayment trends, RFM scores,** and **category-level spending**.

Thank You !