LendingClub Policy Optimization — Deep Learning vs Offline Reinforcement Learning

Author: Aryan Kakkar

1. Introduction

This project investigates how credit approval policies can be optimized using two fundamentally different paradigms — a Deep Learning (DL) classifier trained to predict loan default, and an Offline Reinforcement Learning (RL) agent trained via Conservative Q-Learning (CQL) to maximize long-term profit. The goal is to compare both models, interpret their metrics, and understand the implications of their decisions on business objectives.

2. Our Approach

Step 1 — Data preparation. We start from a LendingClub-like historical dataset that primarily contains approved loans. We clean features (income, DTI, FICO, loan amount, interest rate, term, purpose) and construct the target label "did this loan default?" for supervised learning. For RL we additionally construct a reward signal derived from the loan outcome.

Step 2 — Supervised Deep Learning model. We train a binary classifier to estimate P(default | x). The output is a probability. At inference time, a business rule ("approve if P(default) < τ") converts the probability into an approve/deny policy. This is the classic credit-risk setup.

Step 3 — Offline RL with CQL. Instead of predicting default, we directly learn a policy that chooses an action {deny, approve} to maximize monetary reward. The reward is designed as: 0 for deny, positive for repaid loans (principal + interest), and negative for defaults (loss of principal). Because we only have logged data (no live interaction), we use Conservative Q-Learning (CQL), which keeps the learned policy close to the behavior policy to avoid extrapolation error.

Step 4 — Off-policy evaluation. For the RL policy, we cannot simply compute accuracy. We instead estimate the *policy value* via Fitted Q Evaluation (FQE) / off-policy evaluation to answer: "What would be the expected profit per loan if this policy ran in production?"

Step 5 — Comparison. We finally compare the DL classifier (which optimizes classification metrics) with the RL agent (which optimizes business return). This comparison shows that a good classifier is not always the same as a good policy.

3. Model Results and Key Metrics

Model	Metric(s)	Value	Interpretation
Deep Learning Classifier	AUC (ROC)	~0.707	Reasonably good at ranking good vs bad borrowers; values around 0.70–0.71 are realistic for consumer credit data.
Deep Learning Classifier	F1-Score	~0.5048	Balanced precision–recall on an imbalanced dataset; better indicator than accuracy for LendingClub-style defaults.
RL Agent (CQL)	Estimated Policy Value	-3.3×10^3 Per loan	Learned policy stayed close to the approve-all behavior because the offline dataset had very few examples of the deny action.

4. Why Different Metrics?

For the supervised DL model, the objective is to correctly classify loans into 'will default' vs 'will repay'. Therefore:

• **AUC (Area Under ROC)** ≈ **0.707** ⇒ tells how well the model can rank a random good vs bad borrower. A value around 0.70–0.71 is realistic for retail credit with noisy features. • **F1-Score** ≈ **0.5048** ⇒ balances precision (don't approve bad loans) and recall (don't miss good loans). Because these datasets are imbalanced, F1 is more informative than plain accuracy.

For the **RL agent**, the goal is different: it is not trying to *predict*, it is trying to *decide*. So the right metric is **Estimated Policy Value** (via FQE), which answers: "If we deployed this policy, what would be the average profit per loan?" This directly connects the model to business KPIs.

5. Comparing the Policies

- **DL policy (implicit):** approve if predicted default probability < threshold. You can tune this threshold to trade off approval rate vs bad-loan rate.
- **RL policy (explicit):** agent directly outputs action {0: deny, 1: approve} to maximize reward defined as: 0 for deny, +loan×rate for repaid, –loan for default.

Because the offline dataset only had approved loans (action=1), the RL algorithm saw almost no examples of the deny action. Conservative Q-Learning therefore stayed close to the behavior policy → effectively "approve-all". That is why your RL policy value matched the always-approve baseline.

6. Example: When Do They Disagree?

- **Applicant A:** High income, low DTI, good FICO → DL approves, RL approves.
- **Applicant B:** Medium income, high interest rate, a bit risky → DL may reject because P(default) is high, but RL may approve because (loan_amount × interest_rate) creates a high positive reward in the logged data.
- **Applicant C:** Borderline profile → DL approves (because score is just under the threshold), RL denies (because past loans with similar profile had net negative reward).

This shows: *DL reasons in probability space, RL reasons in profit space.*

7. Limitations & Future Work

Current limitations:

1. **Action coverage problem:** We do not observe outcomes for denied loans → RL cannot learn when denying is better.
2. **Reward realism:** Current reward ignores partial recoveries, servicing costs, charge-off timing, and time value of money.
3. **Offline RL conservativeness:** CQL is conservative by design; with poor coverage it collapses to the behavior policy.

Future work:

- **Collect counterfactual / deny data:** run A/B or contextual-bandit style logging to record some deny actions so the RL agent can see both actions.
- **Richer economic reward:** include recovery_amount, collection_fees, prepayment, and discount future cashflows to present value.
- **Try alternative offline RL algorithms:** IQL, BCQ, CQL with action-augmentation, or decision transformers trained on trajectories.
- **Hybrid policy:** deploy DL as the main scorer, but hand over *only borderline cases* (e.g. P(default) in [0.35, 0.55]) to the RL agent. This reduces risk but still tests RL in production.
- **Feature expansion:** add bureau / alternate data (utility, telecom, GST, bank statements) to improve AUC beyond ~0.71.
- **Causal modeling:** use uplift / causal forests to estimate the effect of approve vs deny, which directly addresses the missing-action problem.

8. Conclusion

The supervised model is currently the safer, more production-ready choice: it performs reasonably well (AUC ≈ 0.707, F1 ≈ 0.5048), it is stable, and it is easy to explain to risk teams. The RL agent, in its current offline setting, shows the right direction (optimize economic value, not accuracy) but needs better data — especially denied-loan outcomes — before deployment. A practical roadmap is: deploy the DL model now, run the RL policy in shadow mode to collect more diverse logged data, and then gradually move to RL-driven approvals once action coverage improves.