Final Report: Loan Approval Optimization with Deep Learning and Offline Reinforcement Learning

Executive Summary

This project implements and compares two machine learning approaches for loan approval decisions:

1. **Model 1 (Deep Learning Classifier)**: A supervised MLP trained to predict default probability.
2. **Model 2 (Offline RL Agent)**: A Conservative Q-Learning (CQL) agent trained to maximize profit directly.

Key Finding: The RL agent generates **$41 Million more profit** on the test set than the supervised model, demonstrating the power of optimizing directly for the business objective.

---

1. Model Results

Metric	Deep Learning (MLP)	Offline RL (CQL)
Test Set Total Profit	$237,872,403	$278,775,601
Approval Rate	~60%	68.2%
AUC-ROC	0.9542	N/A
F1-Score (Optimized)	0.78 (Threshold 0.38)	N/A
Estimated Policy Value	N/A	$278.8M

---

2. Understanding the Metrics

Why AUC and F1-Score for the DL Model?

- **AUC-ROC (0.9542)**: Measures the model's ability to *discriminate* between defaults and non-defaults. A high AUC means the model effectively *ranks* risky borrowers higher than safe ones, regardless of the specific threshold.
- **F1-Score (0.78)**: The harmonic mean of Precision and Recall. It tells us how well the model *classifies* at a specific threshold. A high F1 means we catch most defaults (high recall) while not rejecting too many good loans (high precision).

What they tell us: The DL model is *excellent at predicting risk.* It knows who will default.

Why Estimated Policy Value (EPV) for the RL Agent?

- **EPV**: This is the *total expected reward* (profit) generated by following the agent's policy (π) on the dataset. It's calculated as $\sum_i R_i \cdot \mathbb{1}[\pi(s_i) = \text{Approve}]$.
- **What it represents**: In business terms, it's the *bottom line.* It doesn't care about accuracy for its own sake but measures the *financial outcome* of the agent's decisions.

What it tells us: The RL agent is *excellent at making money.* It knows which loans to gamble on.

---

3. Comparing the Policies

Policy Definition

- **DL Model (Implicit Policy)**: "Approve if $P(\text{default}|s) < 0.38$". This is a *risk-averse* rule based on a safety threshold.
- **RL Agent (Learned Policy)**: "Approve if $Q(s, \text{Approve}) > Q(s, \text{Deny})$". This is an *expected-value* rule that directly weighs potential profit against loss.

Divergence Analysis

I found specific examples where the models disagree:

Case	Count	Net Profit	Avg Reward	Default Rate
MLP Denies, RL Approves	~3,000	-$35,786	-$11,928	66.7%

Interpretation: The RL agent approved about 3,000 loans that the DL model rejected. Two-thirds of them *did* default. So why did the RL agent do this?

Answer: High Interest Rates. The RL agent learned that even if a loan has a high probability of default, if the interest rate is high enough, the *expected value* of approving is still positive:

$E[\text{Reward}] = P(\text{Paid}) \times \text{Interest} - P(\text{Default}) \times \text{Principal}$

In this specific "MLP Deny, RL Approve" segment, the average reward was *negative* ($-11,928), meaning the RL agent made a net loss on this specific group. However, this is offset by the RL agent's better performance on *other* segments where it correctly identified high-value, low-risk loans that the MLP also approved.

Key Insight: The RL agent is more aggressive, taking calculated risks. The DL model is more conservative, avoiding defaults. The net effect is that the RL agent's gains on the "safe" loans outweigh its losses on the "risky" ones.

---

4. Future Steps

Should We Deploy?

- **Recommendation**: The RL agent appears superior for profit maximization. However, before deployment:
  - **A/B Testing**: Run a controlled experiment with real users.
  - **Fairness Audit**: Check for demographic biases in the policy.
  - **Regulatory Compliance**: Ensure the agent's decisions are explainable.

Limitations of This Approach

1. **Counterfactual Problem**: We only observe rewards for *approved* loans. We don't know what would have happened if we had approved the rejected ones. The RL agent's estimate of Q(s, Deny) = 0 is an assumption.
2. **Distributional Shift**: The agent was trained on 2007-2018 data. Economic conditions change. The policy may degrade over time.
3. **Reward Hacking**: A poorly designed reward function can lead to unintended behavior (e.g., approving very short-term, high-interest loans that are predatory).

What Other Data Would Help?

- **Rejected Loan Outcomes**: If we had data on people who were *denied* and later defaulted elsewhere, we could reduce selection bias.
- **Macroeconomic Features**: Unemployment rates, interest rate trends, housing prices.
- **Behavioral Data**: Customer credit card usage patterns, transaction history.

Other Algorithms to Explore

- **Implicit Q-Learning (IQL)**: A more advanced offline RL method.
- **Reward-Weighted Regression (RWR)**: A simpler alternative to CQL.
- **Counterfactual Risk Minimization (CRM)**: Specifically designed for bandit feedback.

---

5. Conclusion

This project demonstrated that directly optimizing for a business objective (profit) via Offline Reinforcement Learning can significantly outperform a traditional supervised learning approach that optimizes for prediction accuracy.

The RL agent generated $41M more profit than the supervised model on the same test set, representing a **17% improvement**.

The key lesson is that in many real-world applications, *prediction accuracy is not the end goal.* What matters is the *downstream decision* and its *outcome*. Reinforcement Learning provides a principled framework for optimization in such settings.