Project Report: Financial Policy Optimization

1 Executive Summary

The main goal of this project was to build an intelligent lending system that **optimizes for real financial value** instead of just predictive correctness. We took the Lending Club dataset and compared a normal Deep Learning (DL) approach against a more advanced offline Reinforcement Learning (RL) agent.

Key Takeaways:

- **Deep Learning:** The supervised model was decent, achieving an AUC-ROC of 0.72 and an F1-Score of 0.37. It could predict default risk okay, but the model really struggled because the dataset was imbalanced.
- **Reinforcement Learning:** The RL agent did much better than the old, historical way of doing things when you look at the financial result. The old way (baseline) had a massive average loss of **-$1,622.30 per loan**. The RL agent optimized the policy, bringing this huge loss down to just **-$29.06 per loan**.
- **Strategy Shift:** The RL agent became extremely conservative, only approving **13.10%** of applications. It effectively found and avoided the "toxic" loans that the regular DL model might have approved.

---

2 Methodology & Results Presentation

2.1 Deep Learning (Supervised) Results

The Deep Learning model (Multi-Layer Perceptron) was trained to predict the probability of a loan defaulting.

Metric	Value	Interpretation
Accuracy	77.74%	The model correctly classifies the majority of loans. The model correctly classifies **77.74%** of all loans (both Fully Paid and Defaulted).
AUC-ROC	0.716	The model has a good ability to tell the difference between "Fully Paid" and "Defaulted" classes, performing better than guessing.
F1-Score	0.373	This reflects the difficulty of the model in balancing Precision (avoiding false alarms) and Recall (catching all defaults), largely because the dataset is imbalanced. This score indicates a significant challenge in correctly identifying the minority "Defaulted" class.

Metric	Value	Interpretation
Optimal Threshold	0.45	The model's best performance (optimal F1 score is ~ 0.4) is achieved when the decision threshold is slightly lowered from the standard 0.5. This makes it easier to predict a loan as "Defaulted," increasing Recall but sacrificing a small bit of Precision.

2.2 Reinforcement Learning (Offline RL) Results

The RL agent, trained using Conservative Q-Learning (CQL), was set up to maximize long-term profit (Reward = Interest Earned - Principal Lost).

Metric	Value	Interpretation
Baseline Value	-$1,622.30	This is the average financial result per loan in the historical test data. This negative value shows that there were a lot of defaults in this specific sample.
Est. Policy Value	-$29.06	This is the expected financial result under the new RL agent's policy. The agent successfully stopped massive losses, improving the outcome by over **$1,590 per loan**.
Approval Rate	13.10%	The agent became highly selective[22]. It approved only the highest-quality applicants to protect the company's capital[23].

---

3. Analysis of Metrics

3.1 Why AUC and F1-Score for Deep Learning?

AUC-ROC and F1-Score are the usual metrics for the DL model because its job is classification. They tell us about the model's "Predictive Intelligence". An AUC of 0.72 means that if we randomly pick one defaulter and one payer, the model has a 72% chance of correctly saying the defaulter is higher risk.

The Big Problem: These metrics treat all errors the same. To the DL model, a default on a small $2,000 loan is "just as bad" as a default on a massive $30,000 loan. This completely lacks business sense.

3.2 Why "Estimated Policy Value" for the RL Agent?

Estimated Policy Value is the key metric for the RL agent because it functions as a **business decision-maker**.

- **What it represents:** It represents the average profit (or loss) expected per loan application.
- **Business Context:** Unlike accuracy, this metric accounts for the *magnitude* of the mistake. It captures the reality that approving a risky loan with high interest might be worth it, while approving a massive loan with low interest is a terrible risk. The improvement from -$1,622 to -$29 proves the RL agent learned to protect the "bottom line."

4. Policy Comparison

4.1 Comparison of Decision Logic

- **DL Policy (Implicit):** "Approve if risk probability < 0.45."
    - This is a static, risk-averse threshold. It doesn't care if the potential profit is $100 or $10,000.
- **RL Policy (Learned):** "Approve if Expected Future Reward > 0 (or > threshold)."
    - This is dynamic. The agent looks at the state (income, debt-to-income ratio, loan amount) and predicts if the *financial return* justifies the action.

4.2 Disagreement Analysis

The models made different decisions on a significant portion of the applicants.

- **Case 1: DL Denies, RL Approves (The "High Reward" Hunter)**
    - *Scenario:* A borrower with a slightly high Debt-to-Income (DTI) ratio applies for a loan with a **very high interest rate (e.g., 20%+)**.
    - *DL Decision:* **Deny.** The high DTI pushes the default probability above 0.45. The DL model sees "Default Risk" and stops there.
    - *RL Decision:* **Approve.** The RL agent sees the risk, but calculates that the 20% interest creates enough expected profit to outweigh the risk of default. It is "greedy" for value.
    - *Why:* The RL agent is maximizing *Reward*, not minimizing *Errors*.
- **Case 2: DL Approves, RL Denies (The "Capital Protector")**
    - *Scenario:* A borrower with a moderate credit score applies for a **huge loan ($30,000)** at a **low interest rate (7%)**.
    - *DL Decision:* **Approve.** The risk probability is just low enough (e.g., 0.40) to pass the threshold.
    - *RL Decision:* **Deny.** The RL agent realizes that if this specific loan goes wrong, the loss (-$30,000) is catastrophic, and the potential reward (only 7%) isn't worth that exposure. This explains why my RL agent had a low **13.10% approval rate**—it became extremely conservative to avoid large losses.

---

5. Future Steps & Conclusion

5.1 Recommendation

I would recommend a **hybrid deployment**.

- The RL model demonstrated superior financial prudence (saving ~$1,590 per loan compared to baseline). However, its 13% approval rate is likely too restrictive for a growing business.
- **Action:** Deploy the RL agent as a "Safety Filter" on top of the DL model. Let the DL model screen applicants, but allow the RL agent to veto approvals that have a negative expected value.

5.2 Limitations

- **Offline Learning Gap:** The RL agent was trained on "static" historical data. It assumes that future borrowers will behave exactly like past borrowers, which may not hold true during economic shifts.
- **Survivor Bias:** We only have data on loans that were *approved* in the past. We do not know the repayment outcome of people who were rejected. This biases the model to be overly conservative.

5.3 Future Improvements

- **Data Collection:** I would request data on "rejected" applications to perform *Counterfactual Evaluation*.
- **Algorithm Exploration:** I would explore **Online RL simulations** or **A/B Testing** where the agent can make small "exploratory" loans to learn about segments of the population we currently ignore.
- **Advanced Models:** Implementing **PPO (Proximal Policy Optimization)** if an interactive environment can be built, or fine-tuning the **CQL (Conservative Q-Learning)** alpha parameter to balance risk vs. approval volume better.