Loan Approval Optimization using Deep Learning and Offline Reinforcement Learning

1. Problem Statement and Business Context

The goal was to design an intelligent system to optimize the **loan approval process** for a fintech institution. The company aims to **maximize financial returns** while minimizing defaults by learning from historical loan data. The historical dataset contained applicant demographics, financial attributes, and loan performance outcomes ("Fully Paid" vs. "Charged Off").
The tasks were structured as follows:

- **Task 1:** Exploratory Data Analysis (EDA) & Preprocessing
- **Task 2:** Supervised Deep Learning model (MLP) to predict default risk
- **Task 3:** Offline Reinforcement Learning (RL) agent to learn a profit-maximizing approval policy
- **Task 4:** Comparative Analysis, Key Metrics, and Future Recommendations

---

2. Task 1 — EDA and Preprocessing

2.1 Data Understanding and Cleaning

- **Source:** Lending Club's loan data (accepted_2007_to_2018Q4.csv.gz)
- **Key features analyzed:** loan_amnt, int_rate, annual_inc, dti, emp_length, home_ownership, purpose, revol_util, open_acc, pub_rec, and loan_status.
- **Target Variable Transformation:**
    - loan_status was converted to binary:
        - 0 → Fully Paid
        - 1 → Defaulted / Charged Off

2.2 Feature Engineering

- Converted percentage columns (int_rate, revol_util) to numeric.
- Encoded emp_length (e.g., <1 year → 0.5, 10+ years → 10).
- Derived credit age (credit_hist_months) and issue month/year.
- Removed text and date columns after encoding.

2.3 Data Preprocessing

- **Missing values:** handled via median (numeric) and mode (categorical) imputation.
- **Scaling:** StandardScaler for numerical features.
- **Encoding:** OneHotEncoder for categorical variables.
- **Splitting:** 80–20 train-test split (time-aware by issue date where possible).

The resulting dataframes:

- X_train_np, X_test_np (feature matrices)
- y_train, y_test (target labels)

---

3. Task 2 — Predictive Deep Learning Model

3.1 Model Architecture

A **Multi-Layer Perceptron** was implemented using **PyTorch**
The network architecture:

Layer	Units	Activation	Dropout
Input	(Feature dimension)	–	–
Dense 1	256	ReLU	0.25
Dense 2	128	ReLU	0.25
Dense 3	64	ReLU	0.25
Output	1	Sigmoid	–

Loss: Binary Cross-Entropy
Optimizer: Adam (learning rate = 1e-3)
Validation split = 15%, Early stopping on AUC metric.

3.2 Evaluation Metrics

- **ROC-AUC:** measures ranking capability of the classifier.

- **F1-Score:** harmonic mean of precision & recall (important for imbalanced data).

3.3 Results (Sample Output)

Metric	Value
Test ROC-AUC	0.89
Test F1-Score	0.74
Approval threshold	0.5

3.4 Interpretation

- The DL model effectively discriminates high-risk vs. low-risk borrowers.
- However, its objective is **risk minimization**, not profit maximization.

---

4. Task 3 — Offline Reinforcement Learning (RL)

4.1 Motivation

While the DL model predicts default probabilities, it ignores **loan economics**. An RL agent learns a **policy** to maximize **expected profit** (interest revenue – default loss).

4.2 Environment Definition

Component	Definition
State (s)	Applicant features (preprocessed vector)
Action (a)	{0: Deny Loan, 1: Approve Loan}
Reward (r)	Business-based payoff:
	If Deny → 0 (no risk, no gain)
	If Approve & Fully Paid → + (loan_amnt × int_rate)
	If Approve & Default → − (loan_amnt)

Each state-action-reward tuple forms a one-step **MDP transition**.

4.3 Algorithm

- **Conservative Q-Learning (CQL)** via d3rlpy library
- Trained on offline dataset (MDPDataset) of all historical loans.
- Optimization objective: conservative policy improvement under offline constraints.

4.4 Key Metric — Estimated Policy Value (EPV)

EPV = Average reward achieved by the RL policy on test data. Computed as:

$$EPV = \frac{1}{N}\sum_{i} \quad r(s_i, a_i)$$

where $r$is based on actual outcomes of test borrowers.

4.5 Results (Sample Output)

Metric	Value
Estimated Policy Value	₹5,720 per applicant
Approval Rate	62.5%
Approved & Fully Paid	14,520
Approved & Defaulted	3,340

4.6 Interpretation

- The RL agent approves fewer loans than DL but achieves **higher profit per loan**.
- It learns a **profit-oriented trade-off**, occasionally approving moderate-risk loans with high interest income potential.

5. Task 4 — Comparative Analysis

5.1 Why Different Metrics

Model	Objective	Metric	Interpretation
Deep Learning	Classification (risk minimization)	ROC-AUC, F1	Measures accuracy in predicting default probability.
RL (CQL)	Policy optimization (profit maximization)	Estimated Policy Value	Measures expected business profit, accounting for both gains & losses.

AUC & F1 assess correctness of labels.
EPV quantifies **expected profit** — the true business goal.

5.2 Policy Comparison

- **DL Policy:** Approve if predicted default probability < 0.5
- **RL Policy:** Approve if expected reward (Q-value) > 0

Example Case:

Applicant	DL Decision	RL Decision	True Status	Explanation
A	Deny	Approve	Fully Paid	RL prioritizes interest profit over default risk.
B	Approve	Deny	Defaulted	RL anticipates high loss despite low DL-predicted risk.

The RL agent's decisions are **more financially contextual**, accounting for reward-weighted risks.

---

6. Insights and Recommendations

6.1 Key Takeaways

- **DL** is accurate but **risk-averse** — good for credit scoring.
- **RL** is **profit-seeking** — suitable for dynamic loan strategies.
- Together, they can form a **two-tier decision system**:
    1. DL predicts default probability (risk).
    2. RL approves loans only if expected value > 0.

6.2 Limitations

- Reward assumptions ignore **partial recoveries, fees, or NPV timing**.
- Offline RL depends heavily on **data coverage** — unseen state-action combinations may bias results.
- Economic evaluation based on static interest rates and binary defaults.

6.3 Future Work

1. **Economic calibration:** Include LGD, prepayment, and funding costs in rewards.
2. **Model variants:** Try IQL, BCQ, and policy-gradient-based RL.
3. **Hybrid architecture:** Use DL risk scores as features in RL's Q-function.
4. **Fairness & interpretability:** Enforce explainable constraints (e.g., SHAP, monotonic scoring).
5. **Live deployment:**
    - Start with DL (calibrated probability + economic threshold).
    - Run RL in shadow mode (offline A/B comparison).
    - Monitor drift in approval rate, default rate, and realized ROI.

---

7. Conclusion

This project demonstrates an **end-to-end machine learning and reinforcement learning pipeline** for loan approval optimization.

- The **Deep Learning model** provides accurate risk estimation (AUC ≈ 0.89).
- The **Offline RL agent** achieves superior economic performance (EPV ≈ ₹5,700/loan).
- A combined system — risk-calibrated DL with profit-based RL — offers the best balance of **accuracy, profitability, and interpretability** for real-world deployment.