Final Report — Policy Optimization for Financial Decision-Making

Author: Ansh Sharma
Date: _____

Executive summary

This project compares a supervised loan-default classifier with an offline RL policy for loan approval decisions. We implemented a reproducible pipeline covering EDA, preprocessing, a PyTorch MLP classifier, offline RL (CQL), and offline policy evaluation (OPE). The supervised model is evaluated on AUC and F1; the RL policy is evaluated via OPE (e.g., FQE). Key deliverables include: code notebooks, saved artifacts (preprocessor and model weights), and this concise report.

1. Dataset

Dataset: Accepted-loans dataset (Lending platform). After filtering to clear outcomes (Fully Paid / Charged Off / Default), the final dataset contains N rows (replace with your run's value). The target is binary: 0 = Fully Paid, 1 = Default/Charged Off. Important limitation: the dataset primarily contains approved loans (no rejected applications), which biases the behavior policy and impacts offline RL and OPE.

2. Preprocessing & Feature Engineering

Selected features include: loan_amnt, int_rate, annual_inc, dti, open_acc, revol_util, income_to_loan_ratio, grade, home_ownership, purpose, emp_length, verification_status (adapt as available). Key steps: convert 'int_rate' strings to numeric (e.g., '13.56%' → 0.1356), create derived feature income_to_loan_ratio, drop rows with missing values in required features (for simplicity), standardize numerical features (StandardScaler), and one-hot encode categorical features (OneHotEncoder with handle_unknown='ignore'). The fitted preprocessor was saved for reproducibility.

3. Supervised Baseline

Model: PyTorch MLP with architecture input → 128 → 64 → 1 (Sigmoid). Training used binary cross-entropy loss, Adam optimizer (lr=1e-3), batch size 256, and 10 epochs (use validation and early stopping for production). Evaluation metrics: AUC (discrimination) and F1 (thresholded performance). Replace the placeholders below with your run results: - AUC = X.XX - Best F1 = Y.YY at threshold T

4. Reward Design & Offline RL Setup

Action space: {0: Deny, 1: Approve}. Reward design (monetary): - Deny (0): reward = 0 - Approve (1) & fully paid: reward = loan_amnt × int_rate (interest revenue) - Approve (1) & default: reward = –loan_amnt (principal loss) We framed episodes as single-step MDPs (each application is one decision). Algorithm: Conservative Q-Learning (CQL) via d3rlpy. Note: because the dataset lacks rejected applications, the historical behavior action is assumed to be approve for available rows; document this limitation in the report.

5. Offline Policy Evaluation & Comparison

OPE methods: Fitted Q Evaluation (FQE) is recommended for lower variance estimates; Doubly Robust (DR) and Importance Sampling (IS) are alternatives. OPE estimates have variance—report confidence intervals and compare multiple estimators. Comparison approach: estimate expected monetary return per application for each policy and inspect disagreement cases where supervised and RL policies differ. Typical patterns: supervised model may deny high-default-risk applicants while RL approves if expected interest exceeds expected loss; RL may be conservative in underrepresented states.

6. Limitations & Future Work

Key limitations: - Dataset bias: lack of rejected applications affects behavior policy and OPE reliability. - Reward simplifications: treats default as total loss and full repayment as immediate revenue; ignores time value of money and partial recoveries. - OPE variance: IS-based estimators can be high variance when target and behavior policies differ. Future work: - Collect application-level data including rejected applications to estimate a realistic behavior policy. - Incorporate loan term discounting and partial recoveries into the reward. Model time-to-default for richer value estimation. - Use constrained RL for risk budgets and fairness constraints; run small-scale randomized A/B tests to validate OPE results.

7. Results & Artifacts (how to reproduce)

Artifacts produced by the pipeline: - Preprocessor (joblib) and processed arrays (numpy .npy files). - Supervised model weights (PyTorch .pth) and predicted probabilities (.npy). - RL policy artifacts (d3rlpy saved policy) and optional OPE outputs. Reproduce steps: 1. Install dependencies from requirements.txt 2. Place dataset CSV under data/ and update DATA_PATH 3. Run notebooks in order: preprocessing → supervised training → RL training & OPE → analysis Expected outputs (replace with your numbers after running the code): - Supervised AUC = X.XX - Supervised best F1 = Y.YY (threshold T) - Estimated policy value (CQL via FQE) = ■Z per application Notes: provide these concrete numbers (AUC, F1, estimated returns) in the final submission after running experiments.

Appendix — Quick answers

- Why AUC & F1? AUC measures ranking ability across thresholds; F1 balances precision and recall at a chosen decision threshold. - Why this reward? Monetary reward aligns the policy objective with business ROI: maximize expected interest revenue while minimizing principal losses. - How to trust OPE? Use multiple estimators (FQE, DR, IS), report confidence intervals, and where possible validate with randomized trials.

Prepared by: Ansh Sharma