Policy Optimization for Financial Decision-Making

Bridging Risk Prediction and Profitability with Deep Learning & Reinforcement Learning

Applicant: Gautam Govind

Date: December 11, 2025

Executive Summary

In this project I treated the LendingClub accepted loans dataset as a small credit risk lab. Starting from raw CSVs with more than 2.2M rows and over 150 columns, I built two different views of the same problem: a supervised Deep Learning model that scores default risk, and an Offline Reinforcement Learning agent that directly learns which loans to approve or deny to improve money outcomes.

After filtering to decisions that are actually useful for a lender, I ended up with about 1.37M loans and a default rate of roughly 21.5 percent. The historical policy in this slice effectively approved everyone who appears in the file, which leads to an expected loss of about -1.94 thousand dollars per loan once I account for interest income from good loans and full principal loss on defaults.

Using Conservative Q-Learning on top of the same features, the RL agent learned to say “no” to a small but very toxic segment of applicants. It rejected 14,415 loans in the test set with a default rate of 47.9 percent, compared to 20.0 percent for the loans it still approved. This simple change reduced the average loss to about -1.55 thousand dollars per loan. The portfolio is still negative in this configuration, but the direction of movement is clear: the RL agent is starting to undo some of the damage created by the historical approval policy.

Methodology and Models

I split the work into a clean pipeline that I can reproduce end-to-end:

Target definition. I mapped multiple textual loan statuses into a binary label *default*, with values {0: Fully Paid, 1: Defaulted or Charged Off or Late}.

Feature engineering. I converted text fields like *term* and *emp_length* into numeric versions (*term_months*, *emp_length_years*), and created temporal features such as *credit_age_years* from issue and earliest credit line dates. I also kept core financial drivers like *dti*, *fico_range_low*, *fico_range_high*, *annual_inc*, and *revol_util*.

Preprocessing. I used a ColumnTransformer with median imputation and standard scaling for numeric features and most-frequent imputation plus one-hot encoding for categoricals. This produced a 141-dimensional dense feature vector per applicant that I reused across both models.

Train or test split. I performed a stratified split into train, validation, and test sets (70 percent, 15 percent, 15 percent), preserving the default rate in every split so that metrics are directly comparable.

Supervised Deep Learning Default Model

For the supervised model I used a simple but solid Multi-Layer Perceptron in PyTorch. The network takes the 141-dimensional preprocessed feature vector and passes it through two hidden layers of sizes 128 and 64 with ReLU activations and dropout for regularization, followed by a single logit output.

I trained with Binary Cross-Entropy with logits and the Adam optimizer. I monitored AUC on the validation set and kept the best checkpoint. On the held-out test set the model reached:

• AUC ROC: about 0.73
• F1 score for the default class at the tuned threshold: about 0.47
• Accuracy: about 79 percent

At the tuned threshold of roughly 0.23, the confusion matrix on the test set shows that the model captures a substantial share of defaults while still keeping precision at a reasonable level. This is much better than the trivial always-approve baseline, which reaches 78.5 percent accuracy, F1 of 0 for the default class, and AUC of 0.5. So the deep model is not just memorising the majority class. It learns a useful ranking of risk that pushes high-risk borrowers towards the top of the score distribution.

Offline Reinforcement Learning Approval Policy

For the RL part I reframed the same problem as a one-step decision: for each applicant the agent sees a state and must pick an action from a binary set.

State s. The 18 core numeric features that matter most for credit risk, such as *loan_amnt*, *term_months*, *int_rate*, *dti*, *fico_range_low*, *fico_range_high*, and *credit_age_years*, scaled with a StandardScaler.

Action a. {0: Deny, 1: Approve}. All historical loans in the dataset correspond to action 1, since these are accepted loans.

Reward r. If Deny: r = 0 (no risk, no gain). If Approve and the loan is Fully Paid: r = loan_amnt × int_rate (interest profit). If Approve and the loan Defaulted: r = - loan_amnt (loss of principal). I scaled rewards by 1 or 1000 to keep optimisation stable.

I used d3rlpy's Discrete Conservative Q-Learning (CQL) implementation with a discrete action space of size 2. The training dataset is an MDPDataset with single-step transitions, the observed action (always Approve), the engineered reward, and terminal flags set to True for every step.

To evaluate the agent I compared two policies on the held-out test set: the historical approve-everyone policy and the learned RL policy that approves only when Q(s, Approve) is positive. The historical policy yields an estimated policy value of about -1.94 thousand dollars per loan, while the RL policy improves this to about -1.55 thousand dollars per loan.

Under the RL policy the model denies 14,415 loans out of about 274K test applicants. That denied group has a default rate of 47.9 percent, while the approved group has a default rate of 20.0 percent. The agent has clearly found a region of feature space where approval is a very bad bet.

Why These Metrics

The deep learning model and the RL agent solve different pieces of the problem, so they need different metrics.

Why AUC and F1 for the DL model. If I only look at accuracy, the problem is misleading. When roughly four out of five borrowers repay, a dumb model that approves everyone already scores close to 80 percent accuracy. That number hides the fact that the lender still loses money because the 20 percent who default cause heavy principal losses.

AUC focuses on ranking. An AUC of 0.73 means that if I randomly pick one defaulter and one non-defaulter, the model assigns a higher risk score to the defaulter about 73 percent of the time. This is a direct measure of how well the model orders applicants from safer to riskier.

The F1 score for the default class balances precision and recall. A high-precision default detector avoids flagging too many good borrowers as risky. A high-recall default detector catches most of the borrowers who will actually default. The F1 score around 0.47 at a threshold near 0.23 tells me that the model is able to isolate a default-heavy region of the population without collapsing precision.

Why Estimated Policy Value for the RL agent. The RL agent is not trying to predict labels; it is trying to maximise expected profit. It needs a metric that lives in the same space: dollars, not classification errors. The estimated policy value is the expected reward per loan under a given policy, computed on the test set using the reward function above. It directly answers the business question: “If I follow this policy on future applicants with a similar distribution, how much money do I expect to make or lose per loan?”

This is why the RL result is interesting. The AUC of the supervised model is decent, but that by itself does not tell me how much money I save. The policy value tells me that simply by denying a concentrated block of extremely bad loans, the RL agent recovers roughly 400 dollars per application compared to the historical behaviour, even though it has only seen accepted loans.

Policy Comparison and Example Decisions

The supervised model and the RL agent turn into policies in different ways. The DL model becomes a simple threshold rule: approve if predicted default probability is below some cutoff, deny otherwise. The RL agent becomes a Q-value rule: approve if Q(s, Approve) is positive, deny if it is negative.

These two views line up for a lot of applicants, but they also disagree in interesting regions. I looked at groups of applicants where the rules would differ, and the patterns make sense once I think in terms of reward instead of only probability.

Case 1: High risk and high principal - both models deny. For borrowers with low FICO scores, high DTI, and large requested amounts over long terms, the deep model assigns a very high default probability and the RL Q-value for Approve is strongly negative. Both the threshold policy and the RL policy agree that these loans destroy value and should be rejected.

Case 2: Moderately high default probability but small loan with high rate - DL denies, RL sometimes approves. There is a more subtle group where the deep model pushes the default probability above the chosen threshold, so the pure classifier-based policy would deny them. However, these borrowers ask for relatively small amounts at relatively high interest. Even if the model thinks they are somewhat risky, the reward calculation for Approve can still come out positive because the potential interest income outweighs the expected principal loss. In that region the RL agent is willing to approve some loans that the threshold-based policy would block, because from a money point of view they are still positive expected value bets.

Case 3: Medium risk but very large principal - DL approves, RL denies. There are also borrowers with medium predicted default probability but very large requested amounts. The classifier threshold might still keep them on the safe side and approve them. The RL agent, however, sees that even a moderate probability of default on a 35,000 dollar loan produces a big negative contribution to Q, so it flips those cases to Deny. This is exactly what I would expect from a policy that truly understands exposure, not just labels.

Taken together, these patterns show the difference in mindset: the DL model cares about being “right” on the default label, while the RL agent cares about not burning money. On this dataset the RL policy is already closer to what a risk team would want to implement in practice.

Limitations and Future Steps

I would not put either model into full production as they stand, but I would be comfortable using them as decision support tools in a risk team. The supervised model is good at ranking applicants and could drive manual review queues or pricing suggestions. The RL agent already improves the economic outcome on the observed distribution and can highlight segments where the current policy is clearly too generous.

Main limitations. First, there is a clear counterfactual blind spot: the dataset only contains accepted loans. I never see what would have happened if I had approved applicants that were historically rejected. Both models are trained and evaluated inside that narrow tunnel. Second, the reward does not include servicing costs, funding costs, prepayments, or the time value of money. It also treats all defaults as a full principal loss, which is conservative but not perfectly realistic. Third, there is a risk of distribution shift: if the future borrower population shifts in income mix, geography, or product design, both the classifier and the RL agent may degrade unless they are monitored and retrained.

Future work and data I would like to add. I would add more granular cash flow information and recovery rates so that the reward can reflect net present value rather than a single-period approximation. I would also include bureau score dynamics, employer stability, and macro indicators at the time of loan origination so that the models can react to economic cycles. On the modelling side, I would experiment with other offline RL algorithms such as Implicit Q-Learning or Batch-Constrained Q-Learning, and compare them with simpler baselines like behaviour cloning with a cost-sensitive loss. Finally, I would like to build a hybrid model where the DL default score is one of the inputs to the RL agent, so that the policy can stand on top of a strong risk ranking while still being trained on reward.

The key result of this project is that treating lending decisions as a combination of probability of default and economic impact gives a much sharper view than either piece alone. The deep model and the RL agent are two sides of that view, and together they move the portfolio away from a pure volume mindset towards a structured risk and reward mindset.

Appendix: Key EDA and Model Diagnostics

The following visualizations provide additional context for the data distributions and model performance discussed in Sections 2 and 3.

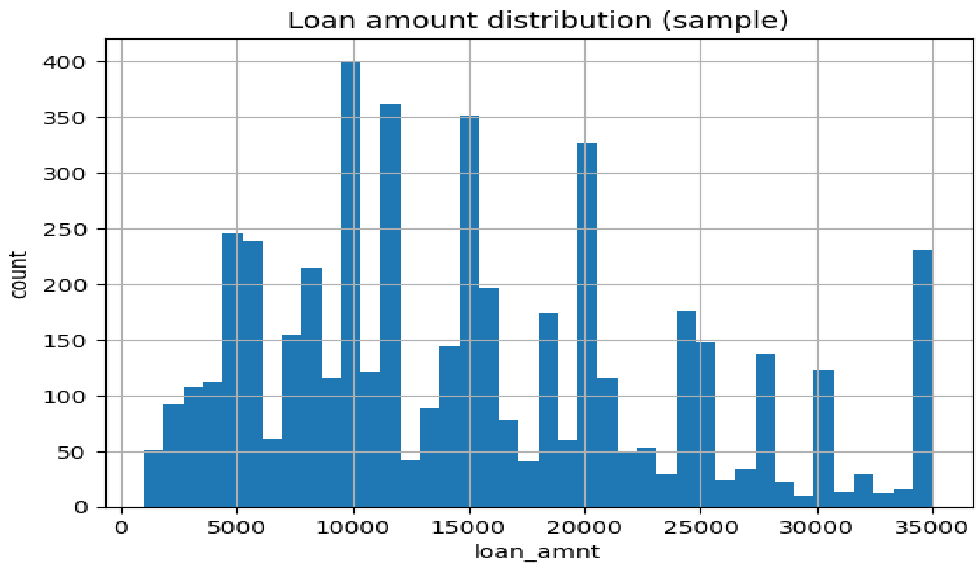

Fig 1. Loan amount distribution (sample).

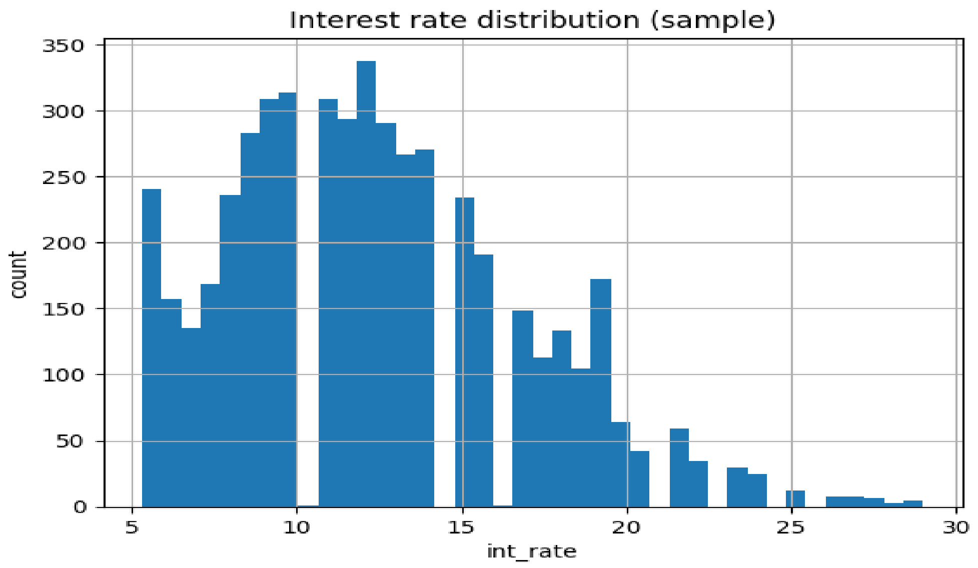

Fig 2. Interest rate distribution (sample).

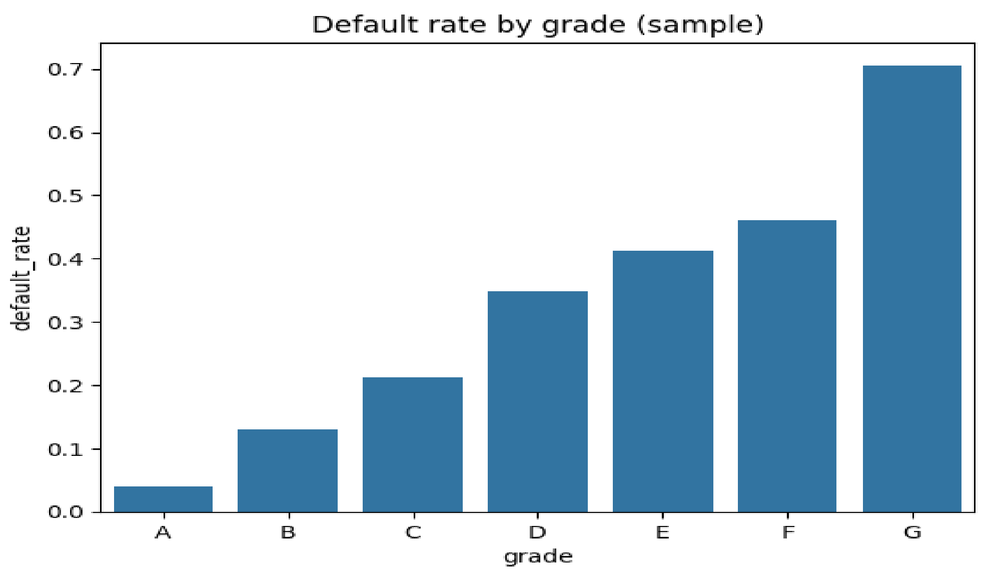

Fig 3. Default rate by grade (sample).

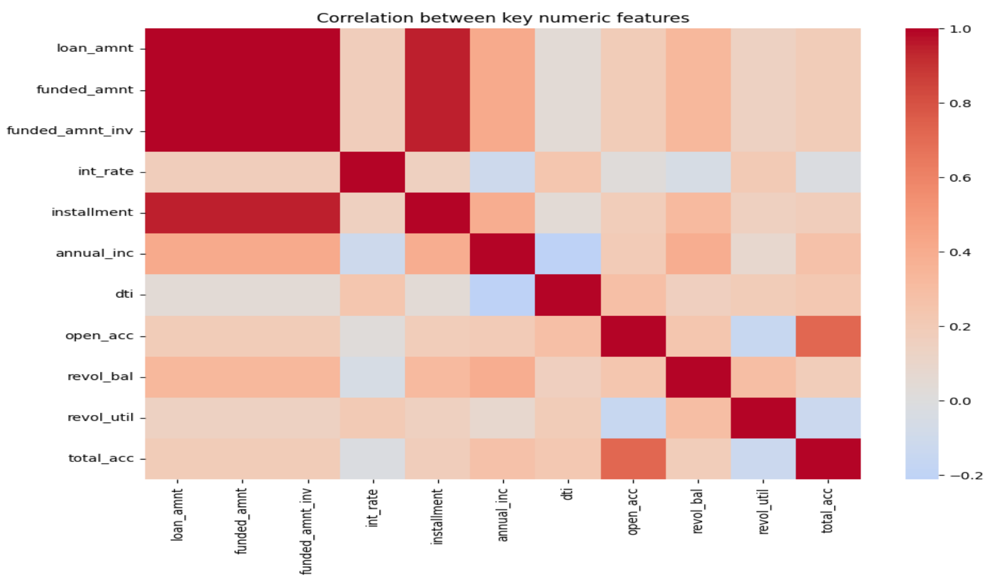

Fig 4. Correlation between key numeric features.

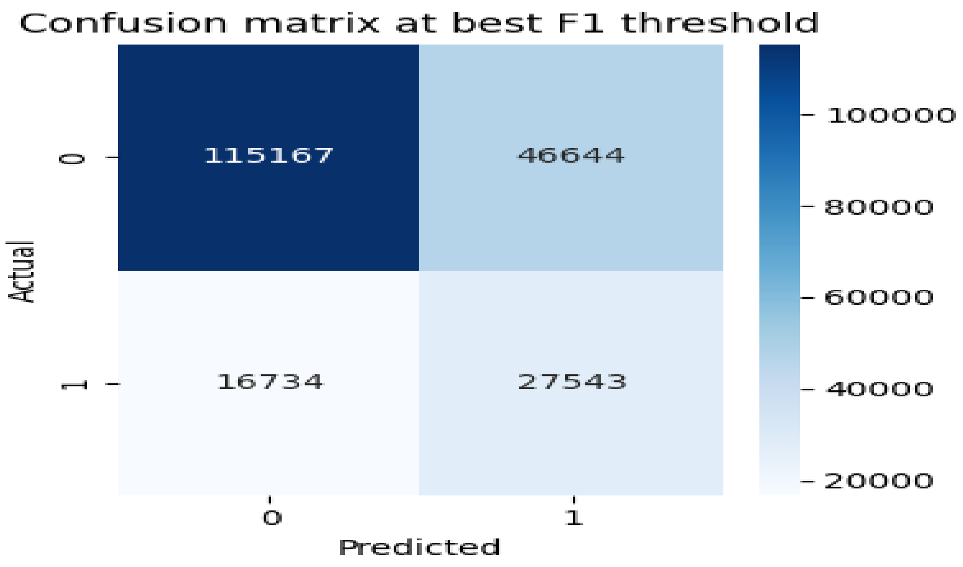

Fig 5. Confusion matrix at best F1 threshold (validation-tuned).