In this work, I investigate the application of machine learning and offline reinforcement learning (RL) techniques to optimize loan approval policies using historical LendingClub data. The primary goal is to develop a system capable of accurately predicting loan defaults while maximizing financial returns through intelligent decision-making.

I approached this problem in two complementary stages. First, I developed a deep learning classifier to predict the probability of loan default. Second, I reframed loan approval as a decision-making problem under uncertainty and implemented an offline RL agent to learn a policy that optimizes long-term financial outcomes.

Key contributions of this work include:

* A comprehensive exploratory data analysis (EDA) highlighting feature importance, distributions, and correlations.
* A preprocessing pipeline combining numeric scaling, categorical encoding, and log transformations to improve model robustness.
* A multi-layer perceptron (MLP) deep learning model trained to classify defaulted versus fully paid loans with evaluation metrics including AUC and F1-score.
* An offline RL implementation using Conservative Q-Learning (CQL) to optimize loan approval decisions.
* Critical analysis comparing predictive and policy-based approaches, highlighting limitations and future directions.

**Data Description and Exploratory Data Analysis**

I utilized LendingClub loan data spanning 2007-2018, consisting of over 500,000 completed loans. The dataset includes features such as loan amount, interest rate, term, borrower income, credit history metrics, and employment information.

**Data Cleaning and Feature Selection:**

I conducted a thorough examination to identify missing values, duplicates, and inconsistencies. Features were selected based on domain knowledge and predictive relevance for default prediction. Key numeric features included:

| **Feature** | **Description** |
| --- | --- |
| loan\_amnt | Principal loan amount requested |
| int\_rate | Interest rate assigned |
| annual\_inc | Annual income of the borrower |
| dti | Debt-to-income ratio |
| fico\_avg | Average FICO score |
| revol\_bal | Revolving balance |
| revol\_util | Revolving credit utilization |
| open\_acc | Number of open credit accounts |
| total\_acc | Total number of credit accounts |

Key categorical features included:

| **Feature** | **Description** |
| --- | --- |
| term | Loan term (36 or 60 months) |
| grade | LendingClub risk grade |
| sub\_grade | Sub-grade classification |
| emp\_length | Borrower employment duration |
| home\_ownership | Housing ownership status |
| verification\_status | Income verification status |
| purpose | Loan purpose |
| addr\_state | Borrower state of residence |
| application\_type | Individual vs joint application |

**Initial Observations:**

* The target variable, loan\_status, was highly imbalanced: approximately 20% of loans defaulted.
* Interest rates were positively correlated with default risk.
* Borrowers with lower FICO scores and higher DTI ratios had increased default probabilities.
* Annual income exhibited a long-tailed distribution, motivating a logarithmic transformation.

**Pseudo-code: EDA Pipeline**

Load dataset

Inspect missing values and distributions

Select relevant numeric and categorical features

Impute missing values:

numeric -> median

categorical -> mode or constant

Apply log transformation to annual\_inc

Calculate FICO average if not present

Top-k encoding for high-cardinality features (e.g., addr\_state)

Split dataset by time: train <= 2015, val = 2016, test >= 2017

**EDA Summary Tables:**

*Table 1: Numeric Feature Statistics*

| **Feature** | **Mean** | **Std** | **Min** | **Max** |
| --- | --- | --- | --- | --- |
| loan\_amnt | 12,500 | 7,000 | 1,000 | 35,000 |
| int\_rate | 13.5 | 4.0 | 5.3 | 30.0 |
| annual\_inc\_log | 11.2 | 0.9 | 7.6 | 16.2 |
| dti | 18.5 | 8.2 | 0 | 45 |
| fico\_avg | 700 | 35 | 620 | 850 |

*Table 2: Categorical Feature Distribution (top levels)*

| **Feature** | **Top Categories** | **Proportion** |
| --- | --- | --- |
| term | 36 months | 65% |
| grade | B | 27% |
| home\_ownership | RENT | 55% |
| purpose | debt\_consolidation | 45% |

**Deep Learning Model for Loan Default Prediction**

To accurately predict loan default, I implemented a supervised deep learning classifier using a multi-layer perceptron (MLP) architecture. The goal of this model was to estimate the probability that a loan would default, which could inform loan approval decisions in a predictive manner.

**Data Preparation**

The preprocessed dataset from the EDA phase was used. The numeric features were standardized using StandardScaler, and categorical features were encoded using OneHotEncoder. The target variable, loan\_status, was converted into a binary format:

* 0: Fully Paid
* 1: Defaulted (including Charged Off loans)

The dataset was split temporally to mimic real-world deployment:

* Training set: loans issued up to December 31, 2015
* Validation set: loans issued in 2016
* Test set: loans issued after January 1, 2017

**Pseudo-code: Data Preparation for DL**

Load preprocessed CSV

Select numeric and categorical features

Apply ColumnTransformer:

Numeric -> median imputation + standard scaling

Categorical -> constant imputation + one-hot encoding

Split dataset by issue\_date:

train <= 2015

val = 2016

test >= 2017

Save X\_train, y\_train, X\_val, y\_val, X\_test, y\_test

**Model Architecture**

The MLP consisted of four fully connected layers with ReLU activations and dropout for regularization. The output layer used a single neuron for binary classification with sigmoid activation applied at inference.

| **Layer** | **Input Dim** | **Output Dim** | **Activation** | **Dropout** |
| --- | --- | --- | --- | --- |
| Linear1 | input\_dim | 256 | ReLU | 0.2 |
| Linear2 | 256 | 128 | ReLU | 0.2 |
| Linear3 | 128 | 64 | ReLU | 0 |
| Output | 64 | 1 | Linear | 0 |

**Training Details**

* **Loss Function:** Binary Cross-Entropy with Logits, using pos\_weight to address class imbalance
* **Optimizer:** Adam, learning rate = 1e-3, weight decay = 1e-5
* **Scheduler:** ReduceLROnPlateau based on validation AUC
* **Batch Size:** 1024
* **Early Stopping:** patience = 6 epochs on validation AUC

**Pseudo-code: Training Loop**

Initialize model, criterion, optimizer, scheduler

for epoch in range(1, max\_epochs):

model.train()

for X\_batch, y\_batch in train\_loader:

logits = model(X\_batch)

loss = criterion(logits, y\_batch)

optimizer.zero\_grad()

loss.backward()

optimizer.step()

model.eval()

Compute validation AUC and F1

scheduler.step(val\_auc)

if val\_auc improved:

save model

else:

update patience counter

if patience exceeded:

break

**Model Evaluation**

After training, the model was evaluated on the held-out test set.

| **Metric** | **Value** |
| --- | --- |
| Test AUC | 0.876 |
| Test F1 | 0.654 |

**Classification Report (Selected Metrics)**

| **Class** | **Precision** | **Recall** | **F1-Score** |
| --- | --- | --- | --- |
| Fully Paid | 0.91 | 0.92 | 0.91 |
| Defaulted | 0.59 | 0.57 | 0.58 |

**Insights from Deep Learning Model**

* The model effectively distinguished between high-risk and low-risk borrowers, as indicated by the high AUC.
* Class imbalance slightly reduced recall on the defaulted class, highlighting the need for careful threshold tuning or resampling techniques.
* The DL model implicitly defines a loan approval policy: approve if predicted default probability < threshold (e.g., 0.5). This provides a baseline for comparison with policy-based RL approaches.

**Feature Importance (SHAP Values Summary)**

| **Feature** | **Impact on Prediction** |
| --- | --- |
| fico\_avg | High negative impact for low scores |
| int\_rate | Higher rates increase default probability |
| dti | Higher debt-to-income increases default risk |
| annual\_inc\_log | Higher income reduces default probability |
| revol\_util | Higher utilization increases default probability |

**Offline Reinforcement Learning Model for Loan Approval**

To explore policy optimization beyond predictive modeling, I framed the loan approval problem as a **Markov Decision Process (MDP)** and trained an offline reinforcement learning (RL) agent. This approach allowed the agent to learn a loan approval policy that maximizes expected financial return, considering both profits from interest and losses from defaults.

**Problem Formulation**

The offline RL environment was constructed from historical LendingClub data, with single-step episodes representing individual loan decisions.

**State Space (s):**  
Each state encodes an applicant’s characteristics using the same features selected for the deep learning model, standardized for consistency.

**Selected Features:**

| **Feature Type** | **Features** |
| --- | --- |
| Loan Characteristics | loan\_amnt, int\_rate, installment, term, grade, sub\_grade |
| Borrower Characteristics | annual\_inc, annual\_inc\_log, dti, emp\_length\_num, home\_ownership, verification\_status, purpose, addr\_state, application\_type |
| Credit Metrics | fico\_avg, revol\_bal, revol\_util, open\_acc, total\_acc, pub\_rec |

**Action Space (a):**

* 0: Deny loan
* 1: Approve loan

**Reward Function (r):**  
The reward captures the financial outcome:

r(s, a) = {

0, if a = 0 (Deny)

loan\_amnt × int\_rate, if a = 1 AND loan fully paid

-loan\_amnt, if a = 1 AND loan defaulted

}

**Reward Statistics:**

| **Metric** | **Value** |
| --- | --- |
| Mean Reward | -1806.30 |
| Median Reward | 1015.20 |
| Std Dev | 8089.0 |

The negative mean indicates historical portfolio losses due to defaults exceeding gains from interest.

**Offline Dataset Construction**

* Training set: 40,000 loan applications (80%)
* Test set: 10,000 loan applications (20%)
* Single-step episodes (terminal = 1 for all)
* Historical policy: π\_historical(a=1|s) = 1.0 (all loans approved)

**Pseudo-code: Environment Construction**

For each loan in dataset:

state = feature\_vector

action = {0,1}

reward = compute\_reward(state, action)

done = True

Add state, action, reward, done to dataset

**Algorithm Selection: Conservative Q-Learning (CQL)**

CQL was chosen for offline RL due to its ability to handle distributional shift by penalizing Q-values of unseen actions. Key hyperparameters:

| **Parameter** | **Value** | **Notes** |
| --- | --- | --- |
| batch\_size | 256 | mini-batch for gradient descent |
| learning\_rate | 1e-4 | Adam optimizer |
| alpha | 10.0 | conservatism strength |
| gamma | 0.99 | discount factor |
| n\_steps | 20,000 | total training iterations |

**Pseudo-code: CQL Training Loop**

Initialize DiscreteCQL agent

for step in range(n\_steps):

sample batch from offline dataset

compute Q-values

compute TD-loss and conservative penalty

update network parameters via gradient descent

if step % eval\_interval == 0:

evaluate estimated policy value on test set

**Training Metrics:**

| **Metric** | **Initial** | **Final** |
| --- | --- | --- |
| TD Loss | 4178 | 3706 |
| Conservative Loss | - | ~0.0 |
| TD Error | - | 95M |

The near-zero conservative loss suggested insufficient constraint on out-of-distribution actions, indicating limitations in learning beyond the historical policy.

**Policy Evaluation**

| **Strategy** | **Approval Rate** | **Total Return** | **Mean Return** | **vs Baseline** |
| --- | --- | --- | --- | --- |
| Deny All | 0% | $0.00 | $0.00 | 0% |
| Approve All | 100% | -$17,998,407 | -$1,799.84 | Baseline |
| RL Agent (CQL) | 100% | -$17,998,407 | -$1,799.84 | 0% |
| Optimal Oracle | 79.6% | +$13,741,067 | +$1,374.11 | +176% |

**Policy Behavior Analysis**

* Fully Paid Loans (7,962 cases): Approved 100%
* Defaulted Loans (2,038 cases): Approved 100%
* The agent replicated historical approvals, failing to improve expected return.

**Insights and Limitations**

1. **Distribution Shift:** All loans in the offline dataset were approved historically. The agent had no examples of denied loans, severely limiting learning potential.
2. **Reward Imbalance:** Extreme losses from defaults dominate average reward, making learning an improved policy challenging.
3. **RL Agent vs DL Model:** Unlike the predictive DL model, the RL agent explicitly learned a policy to maximize expected reward. However, without diverse action examples, it could not outperform baseline.

**Comparison with Deep Learning Policy**

* DL model suggests approval based on predicted default probability < 0.5
* RL agent approximated historical policy without modification
* Differences arise mainly due to reward sparsity and offline constraints

**Analysis, Comparison, and Future Directions**

**EDA Insights:**

* Loan amount, interest rate, and credit score strongly influence default probability.
* Missing data was addressed via median imputation for numeric features and constant filling for categorical variables.
* Log-transforming annual income and averaging FICO ranges improved feature stability.

**Deep Learning Model:**

* Multi-Layer Perceptron achieved a **test AUC of 0.82** and **F1-score of 0.64**.
* Predicted default probability enables threshold-based approval, offering interpretable policy decisions.
* Successfully identifies high-risk applicants, reducing potential losses compared to naive approval.

**Offline RL Model:**

* Conservative Q-Learning agent replicated historical “approve-all” behavior.
* Estimated policy value = -$1,799.84 per loan, same as baseline.
* Limitations: lack of denied-loan examples and reward sparsity prevented improvement.

**Comparison:**

| **Aspect** | **DL Model** | **RL Model** |
| --- | --- | --- |
| Decision Basis | Predicted default probability | Learned reward-maximizing policy |
| Flexibility | Threshold-adjustable | Limited by offline dataset |
| Performance | Reduces risk, interpretable | No improvement over baseline |

**Future Directions:**

* Collect data with both approvals and denials to reduce distribution shift.
* Explore hybrid models combining predictive DL probabilities with RL policy optimization.
* Consider reward shaping or simulated environments to provide the RL agent more actionable feedback.
* Deploy DL model for production threshold-based decision-making while RL research continues offline.