Loan Approval Models: Deep Learning vs. Reinforcement Learning

Karandeep Singh

October 26, 2025

Contents

1 Introduction

- We need to train a model that correctly classifies whether it is profitable to approve the loan request of an individual. This problem is modeled as a binary classification problem.
- We use both the deep learning approach and the reinforcement learning approach to create our models.
- The initial unprocessed data had 152 columns and 2.2 million rows.

2 Data Preprocessing

The raw Lending Club dataset contains a large number of features, including borrower information, loan characteristics, and repayment status. To prepare the data for modeling, several preprocessing steps were applied to clean, reduce, and structure the dataset:

- **Removal of uninformative columns:** Columns that contained only a single unique value or were entirely empty were removed, as they provide no predictive information.
- **Handling missing data:** Columns where more than 60% of values were missing (NaN) were dropped to reduce sparsity and avoid introducing noise into the models.
- **Categorical features with high cardinality:** Columns with too many unique categorical values were removed, as they can lead to overfitting and make one-hot encoding computationally expensive.
- **Subgrade column simplification:** The 'subgrade' column, which contained granular ratings such as 'A1', 'B3', etc., was converted to numeric values to simplify the information, since the coarser 'grade' column was already present.
- **Date feature transformation:** All date columns were split into separate *year* and *month* columns to allow the models to capture temporal patterns without relying on raw datetime strings.
- **Correlation-based feature reduction:** Highly correlated numerical columns with a correlation coefficient greater than 0.9 were removed to reduce redundancy and multicollinearity.

After completing these steps, the dataset was significantly reduced in dimensionality, leaving only 79 informative columns. This cleaned and structured dataset provides a robust input for both deep learning and reinforcement learning models, improving training efficiency and predictive performance.

3 Methodology

3.1 Deep Learning Model

- **Architecture**: Fully-connected neural network with input, hidden layers, and output.
  - Input layer: dimension equals total feature count after preprocessing
  - Hidden layers: 64 neurons (ReLU) $\rightarrow$ 32 neurons (ReLU)
  - Output layer: 2 neurons for binary classification
- **Input preprocessing**: scaling numeric features, one-hot encoding categorical features:
  - Numeric features: StandardScaler normalization, median imputation for missing values
  - Categorical features: One-hot encoding with sparse output disabled, "missing" category for NaN values
  - Features with ≤ 20 unique values treated as categorical
- **Loss function, optimizer, and training procedure**:
  - Loss function: Cross-entropy loss
  - Optimizer: Adam with learning rate 0.001
  - Training: Mini-batch gradient descent with batch size 256
  - Train/test split: 80/20 with stratification
  - Default training: 5 epochs
- **Implicit policy**: approve if predicted default probability $<$ threshold.
  - Probability computed via softmax on output layer
  - Class prediction: $\arg\max(\text{softmax}(\text{output}))$
  - Evaluation metrics: Accuracy, F1-score, and AUC-ROC

3.2 Reinforcement Learning Agent

- Q-network architecture and action space (approve/deny).
  - **State representation**: 78-dimensional feature vector after preprocessing (numeric scaling and categorical one-hot encoding)
  - **Action space**: Binary discrete actions $\{0, 1\}$ where 0 = Deny, 1 = Approve
  - **Network architecture**: Fully-connected neural network
    - Input layer: 78 dimensions (preprocessed features)
    - Hidden layer 1: 128 neurons with ReLU activation
    - Hidden layer 2: 64 neurons with ReLU activation
    - Output layer: 2 neurons producing Q-values $Q(s, \text{Deny})$ and $Q(s, \text{Approve})$
  - **Function approximation**: Neural network approximates $Q(s, a; \theta)$ for continuous high-dimensional state space
  - **Problem type**: Contextual bandit / single-step decision problem (no sequential state transitions)
- Offline RL setup with reward function:
  - Reward for approved loans that are fully paid: $R = \text{loan\_amnt} \times \text{int\_rate}$ (profit from interest)
  - Penalty for approved loans that default: $R = -\text{loan\_amnt}$ (capital loss)
  - Reward for denied loans: $R = 0$ (no financial impact, no risk)
  - **Reward scaling**: Division by 10,000 for numerical stability during training
  - **Offline learning**: Agent trained on static historical dataset without environment interaction
  - Training samples: (s, a, r) tuples where s = applicant features, a = historical action, r = computed reward
- Training procedure and policy evaluation.
  - **Loss function**: Mean Squared Error $L(\theta) = E[(Q(s, a; \theta) - r)^2]$
  - **Optimizer**: Adam with learning rate $\alpha = 0.001$
  - **Training strategy**: Supervised learning on historical data with observed rewards as targets
  - **Batch training**: Mini-batch gradient descent with batch size 256
  - **Data split**: 80% training, 20% testing with random stratification
  - **Epochs**: 10 training epochs (configurable)
  - **Policy at inference**: Greedy policy $a^* = \arg\max_a Q(s, a; \theta)$
  - **Evaluation metrics**:
    - Estimated Policy Value (EPV): Mean reward per decision on test set
    - Total cumulative reward: Sum of all rewards on test data
    - Action distribution: Approval ratio vs. denial ratio
    - Financial outcome: Total profit/loss comparison
  - **Advantage over classification**: Direct optimization of financial outcomes rather than prediction accuracy

4 Results

4.1 Deep Learning Model Metrics

- Accuracy: **99.71**
- F1-Score: **0.9982**
- AUC: **0.9996**

4.2 Reinforcement Learning Agent Metrics

- Estimated Policy Value: **145606.9219**
- Approve/Deny distribution: Approve **80.57**%, Deny **19.43**%

4.3 Comparison of Policies

- **Examples of differing decisions:**
- **Analysis of high-risk applicants approved by RL but denied by DL:**
  - The RL agent is *reward-driven*, learning a policy that maximizes cumulative profit rather than just minimizing default probability.
  - In cases where an applicant appears risky but the potential financial reward is high, the RL agent may take a calculated risk by approving the loan.
  - Conversely, the DL model strictly follows the predicted default probability, leading to more conservative decisions that may deny potentially profitable loans.

4.4 Analysis of Metrics

From the metrics reported above, several insights can be drawn:

- **Deep Learning Model Performance:**
  - The DL model demonstrates extremely high predictive performance with an accuracy of 99.71%, F1-score of 0.9982, and an AUC of 0.9996.
  - These metrics indicate that the model is highly effective at distinguishing between good and bad loan applicants.
  - However, the high accuracy may also suggest potential class imbalance in the dataset, which should be interpreted with caution.
- **Reinforcement Learning Agent Performance:**
  - The estimated policy value of 145,606.92 indicates the expected cumulative reward (profit) that the RL agent achieves by following its learned policy.
  - The agent approves 80.57% of applications, which is higher than the typical approval rate of a conservative DL model. This demonstrates a more risk-tolerant strategy aimed at maximizing reward rather than strictly minimizing defaults.

- **Comparative Insights:**
  - While the DL model prioritizes accuracy and risk avoidance, the RL agent balances risk and reward, allowing some high-risk, high-reward loans to be approved.
  - This is reflected in the differing decisions, where the RL agent may approve loans denied by the DL model, contributing to higher potential profits.
  - The metrics suggest that using RL could lead to better financial outcomes in a profit-driven context, even if it occasionally accepts higher-risk applicants.

5 Discussion

5.1 Why Metrics Matter

- **Deep Learning Model:** F1-Score captures balance between precision and recall, AUC measures discriminative capability.
- **Reinforcement Learning Agent:** Estimated Policy Value directly represents expected financial outcome of the policy. Approve and Deny percentage tell us whether the model is cautious or more risk taking.

5.2 Limitations

- Dataset size and representativeness.
- Assumptions in reward function.
- Potential overfitting of DL model or RL Q-network.
- Dataset Preprocessing Errors.

5.3 Future Steps

- Designing a better reward function
- Using better data preprocessing techniques
- Usage of ensemble methods
- Using Online RL for realtime predictions which would server as a better deployable model