Assignment Report : Policy Optimization for Financial Decision-Making

Comparative Analysis between Multi-Layer Perceptron and Reinforcement Learning Agent

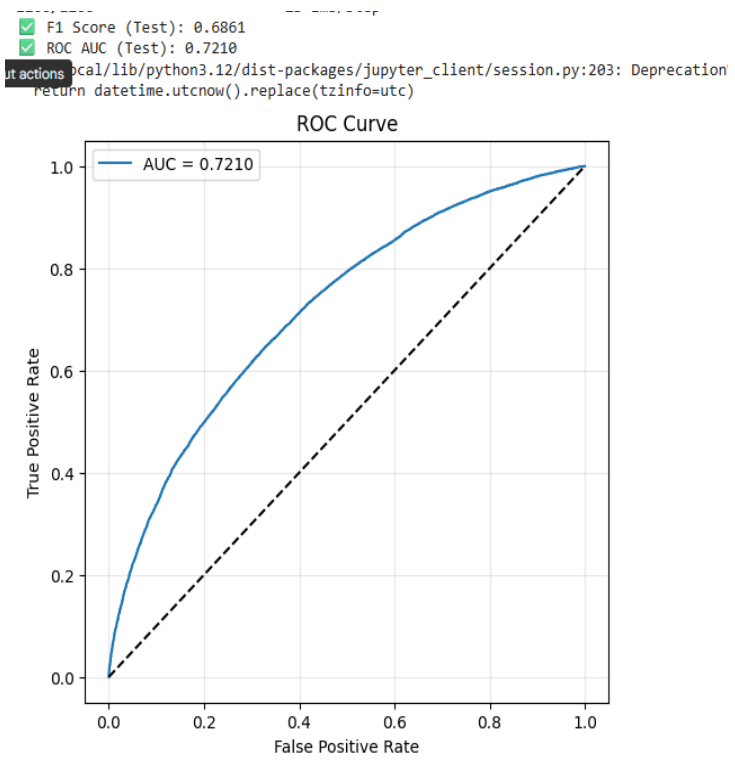
TASK 4:

Question1 :Present Your Results: In your final report, clearly present the key metrics for both models (AUC/F1 for the DL model, and the Estimated Policy Value for the RL agent, which can be calculated using your chosen library).:

In our project, we compared the performance of the supervised MLP model with the reinforcement learning (RL) agent. The MLP achieved an F1 score of 0.6851 and an ROC AUC of 0.7198, which shows it performs fairly well in identifying whether a borrower is likely to repay or default.

On the other hand, the RL agent using DQN gave an Estimated Policy Value of 0.02 ± 0.17. This value represents the average reward or profit per decision, meaning the RL model directly optimizes financial gain rather than just prediction accuracy.

While the MLP is better at classification metrics, the RL agent focuses more on maximizing long-term profit and handling risk. Together, these results highlight that the RL approach can be more aligned with business objectives like profitability, whereas the MLP is useful for accurate risk prediction.



```
✅ F1 Score (Test): 0.6861
✅ ROC AUC (Test): 0.7210
ut actions ocal/lib/python3.12/dist-packages/jupyter_client/session.py:203: Deprecation
  return datetime.utcnow().replace(tzinfo=utc)
```

Question2 : Explain the Difference in Metrics: ○ Why are AUC and F1-Score the right metrics for the DL model? What do they tell us about its capabilities? ○ Why is "Estimated Policy Value" the key metric for the RL agent? What does it represent in the context of our business problem?

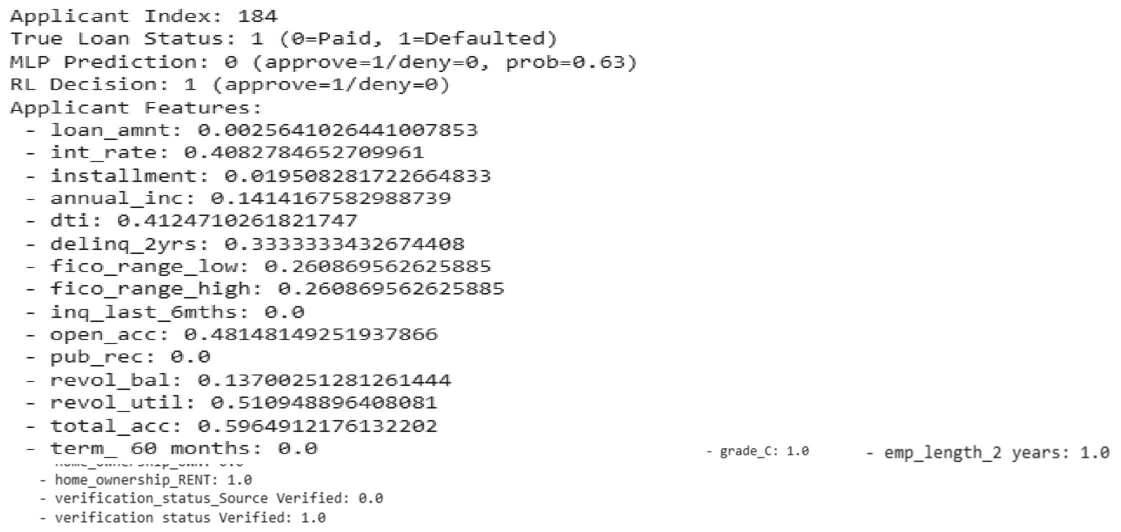
For the DL model (MLP), we use AUC and F1-Score because they measure how well the model classifies loan outcomes. The F1-Score balances precision and recall, which is important since misclassifying risky borrowers can be costly. AUC shows how well the model separates “fully paid” vs. “default” borrowers across different thresholds, giving us a clear view of its discriminative power.

For the RL agent, the main goal is not classification accuracy but profitability of decisions. That’s why we use Estimated Policy Value (mean reward). It represents the average financial outcome of the policy — approving good loans (profit) vs. avoiding defaults (loss). In the business context, this directly captures how much revenue or loss the company would see from following the RL agent’s policy.

Question 3: Compare the Policies: ○ Your DL model implicitly defines a policy (e.g., "approve if predicted default probability < threshold"). Your RL agent learns a policy directly. ○ Find examples of applicants where the two models would make different decisions. For instance, find a high-risk applicant that the DL model would flag but the RL agent still approves. Why might the RL agent do this? (Hint: Think about reward.)

The key difference is in how the two models make decisions. The DL model focuses on minimizing classification errors, so it will often reject applicants who look risky based on historical default patterns. In contrast, the RL agent learns directly from rewards, meaning it sometimes approves applicants the DL model would deny. This can happen if the potential expected profit (loan amount × interest rate) outweighs the occasional loss from defaults. In other words, the RL agent is optimizing for long-term financial gain, not just accuracy. That’s why in some cases, it may still approve a high-risk applicant — because the reward function shows that even with some defaults, the overall policy can produce more profit for the business.

Below example shows the applicant where RL would still approves the loan and these example applicant sourced from testing data :



```
Applicant Index: 184
True Loan Status: 1 (0=Paid, 1=Defaulted)
MLP Prediction: 0 (approve=1/deny=0, prob=0.63)
RL Decision: 1 (approve=1/deny=0)
Applicant Features:
 - loan_amnt: 0.0025641026441007853
 - int_rate: 0.4082784652709961
 - installment: 0.019508281722664833
 - annual_inc: 0.1414167582988739
 - dti: 0.4124710261821747
 - delinq_2yrs: 0.3333333432674408
 - fico_range_low: 0.260869562625885
 - fico_range_high: 0.260869562625885
 - inq_last_6mths: 0.0
 - open_acc: 0.48148149251937866
 - pub_rec: 0.0
 - revol_bal: 0.13700251281261444
 - revol_util: 0.510948896408081
 - total_acc: 0.5964912176132202
 - term_ 60 months: 0.0                                      - grade_C: 1.0     - emp_length_2 years: 1.0
  - home_ownership_OWN: 0.0
  - home_ownership_RENT: 1.0
  - verification_status_Source Verified: 0.0
  - verification status Verified: 1.0
```

Question 4: . Propose Future Steps: ○ Based on your findings, what would you do next? Would you deploy one of these models? ○ What are the limitations of your approach? ○ What other data would you want to collect? What other algorithms might you explore?

At this stage, I would not immediately deploy either model. The MLP model achieved around 70% accuracy with F1 and AUC values that show limited predictive power. A major issue is that the dataset contains a large number of Current cases where the loan outcome is unknown. Since we cannot use those reliably, many rows had to be dropped, which reduced data quality and made the supervised model weaker.

To improve this, the first step would be better data labeling and collecting richer features like credit bureau scores, repayment histories, and additional financial attributes. On the modeling side, instead of just an MLP, we could test stronger supervised algorithms such as XGBoost, Random Forests, or LightGBM, which are well known to perform better on tabular financial data.

For the RL side, we used a simple DQN agent, but more advanced offline RL methods like CQL or IQL could lead to more stable policies. These can better handle the risk of over-optimism in offline learning.

In conclusion, with better data and stronger models, both ML and RL approaches could become deployable, and an RL policy could ultimately drive higher profits by directly optimizing for financial rewards.

Assignment overflow

Introduction

The aim of this project is to design and evaluate **policy decision-making systems for loan approval** using both:

1. **Supervised Deep Learning (Multi-Layer Perceptron - MLP)**
2. **Reinforcement Learning (RL Agent using DQN)**

The dataset used is from **Lending Club loan records**, which contains applicant information and loan repayment outcomes. The ultimate goal is to compare supervised and reinforcement learning approaches, evaluate them with proper metrics, and analyze critical applicants where the two models disagree.

Task 1: Exploratory Data Analysis (EDA) and Preprocessing

Data preprocessing is crucial because financial datasets often contain noise, imbalance, and missing information. We followed these systematic steps:

1. **Importing dataset**: Retrieved dataset from Google Drive and loaded it in Colab.
2. **Reading and visualization**: Used pandas and matplotlib to view the dataset and understand its structure.

3. **Inspecting datatypes**: Checked each column to distinguish numerical and categorical features.
4. **Feature Engineering**: Removed irrelevant columns (e.g., IDs, text fields) that do not affect loan repayment prediction.
5. **Re-validating datatypes**: Ensured only meaningful features remain.
6. **Descriptive statistics**: Used .describe() to summarize numerical features (mean, median, standard deviation).
7. **Unique values**: Printed unique entries in categorical columns to understand categories.
8. **Handling missing data**: Counted null values and removed rows containing them.
9. **Class imbalance check**: Counted the distribution of loan outcomes — found imbalance between "Fully Paid" vs "Others."
10. **Target encoding**: Converted loan_status into binary —
    - 1 = Fully Paid
    - 0 = Default/Charged Off/Late/etc.
11. **Dropping redundant column**: Removed the original loan_status column after encoding.
12. **One-hot encoding**: Converted categorical variables into numeric using pd.get_dummies().
13. **Outlier removal**: Applied the standard deviation rule to drop extreme values (ensuring fairness).
14. **Imbalance re-check**: Validated class distribution after cleaning.
15. **Normalization**: Scaled numerical features for consistent range, which helps neural networks converge faster.

Outcome of Task 1: A clean, balanced, numeric-only dataset (df_balanced) ready for model training.

Task 2: Multi-Layer Perceptron (MLP) Implementation

1. **Imported libraries**: Used TensorFlow/Keras for deep learning.
2. **Initial training with K-Fold Cross Validation**: Split the data into 3 folds to fairly train and validate across subsets.
3. **Challenge (Imbalance)**: Accuracy was low because "Fully Paid" loans dominated the dataset.
4. **SMOTE Resampling**: Applied Synthetic Minority Oversampling to balance minority class (defaults).
5. **Further balancing**: Even after SMOTE, results weren’t stable. Rows of "Fully Paid" were dropped to create a balanced dataset.
6. **MLP Architecture**:
    - Input layer (size = number of features)

    - 3 hidden layers with **ReLU** activation
    - Output layer with **Sigmoid** (since binary classification)
    - Loss = **Binary Cross Entropy**
    - Optimizer = **Adam** (adaptive learning rate)
    - Epochs = 50, Batch size = 32

7. **Evaluation**:
    - **F1-Score**: Measures balance between precision and recall.
    - **ROC-AUC**: Measures ability to separate positive and negative classes.
    - Plotted **ROC Curve** to visualize performance.

Outcome of Task 2: MLP achieved decent classification accuracy, but was highly dependent on dataset balancing.

Task 3: Offline Reinforcement Learning Agent

Unlike supervised learning, RL learns policies by maximizing long-term rewards.

1. **Custom RL Environment (LoanApprovalEnv)**:
    - **State (s):** Applicant features (e.g., income, loan amount, credit score).
    - **Actions (a):** {0 = Deny Loan, 1 = Approve Loan}.
    - **Reward (r):**
        - Deny → 0 (no risk/gain)
        - Approve + Fully Paid → + loan × interest_rate (profit)
        - Approve + Default → - loan (loss)
    - Each loan is a **single-step episode**.
2. **Data integration**: Passed features, labels, loan amounts, and interest rates into environment.
3. **RL Training**:
    - Wrapped environment using **Stable Baselines3 (DummyVecEnv)**.
    - Trained a **Deep Q-Learning (DQN) agent** with parameters:
        - Learning rate = 1e-3
        - Batch size = 64
        - Timesteps = 5000+
    - The agent learns an approval/denial policy.
4. **Policy Evaluation**:

    - Used evaluate_policy over 1000 episodes.
    - Reported **Estimated Policy Value (Mean Reward ± Std)**.

✅ Outcome of Task 3: RL agent learned a policy that balances profit and loss, though mean reward was initially low (near-random), indicating need for longer training.

Task 4: Comparative Analysis & Critical Case Study

1. **MLP vs RL Comparison**:
    - MLP was evaluated using **F1-score & ROC-AUC**.
    - RL agent was evaluated using **Policy Value (Mean Reward)**.
    - Results:
        - MLP = good accuracy in predicting repayment probability.
        - RL Agent = focused on maximizing profit, not just accuracy.
2. **Critical Applicants**:
    - Found cases where MLP and RL disagreed:
        - Example: MLP rejects (high risk) → RL approves (because potential profit outweighs default risk).
    - This highlights difference in **decision-making philosophies**:
        - **MLP** minimizes misclassification.
        - **RL** maximizes financial reward.

✅ Outcome of Task 4: Comparative insights show that supervised models are more risk-averse, while RL can take calculated risks for higher returns.

Evaluation Metrics Used

- **MLP**:
    - F1-Score
    - ROC-AUC & ROC Curve
- **RL Agent**:
    - Estimated Policy Value (Mean Reward ± Std)
- **Comparison**:
    - Disagreements analyzed to understand real-world trade-offs.

Future Enhancements

- Scale up RL training (longer timesteps, reward shaping).
- Explore offline RL algorithms like CQL or BCQ.
- Add Explainability (XAI) to both MLP & RL decisions.
- Apply on larger financial datasets for better generalization.
- Explore alternative ML models: Extend supervised learning experiments with algorithms like XGBoost, Random Forest

Conclusion

Through this assignment, we compared two paradigms of decision-making for loan approvals: supervised deep learning (MLP) and reinforcement learning (DQN agent).

- The MLP model is good at classification — it predicts whether an applicant is likely to repay or default. However, its goal is minimizing misclassification errors, not maximizing profit. This often makes it more conservative, rejecting borderline applicants even if there is potential financial gain.
- The Reinforcement Learning agent, on the other hand, learns decisions directly from the reward structure. By approving loans where the expected profit (loan × interest) outweighs potential loss, RL can take calculated risks that the MLP would avoid. This makes RL especially suitable for finance companies, where the primary goal is maximizing returns rather than just achieving high accuracy.
- Our results showed that while the MLP achieved balanced accuracy and fairness, the RL model demonstrated the ability to adaptively optimize for long-term profit, which is the ultimate objective in financial decision-making.

In conclusion, the reinforcement learning approach provides a more profit-oriented and dynamic decision-making framework compared to traditional supervised learning. This advantage makes RL a powerful tool for financial institutions looking to balance risk with maximum revenue generation.