NN PROJECT REPORT

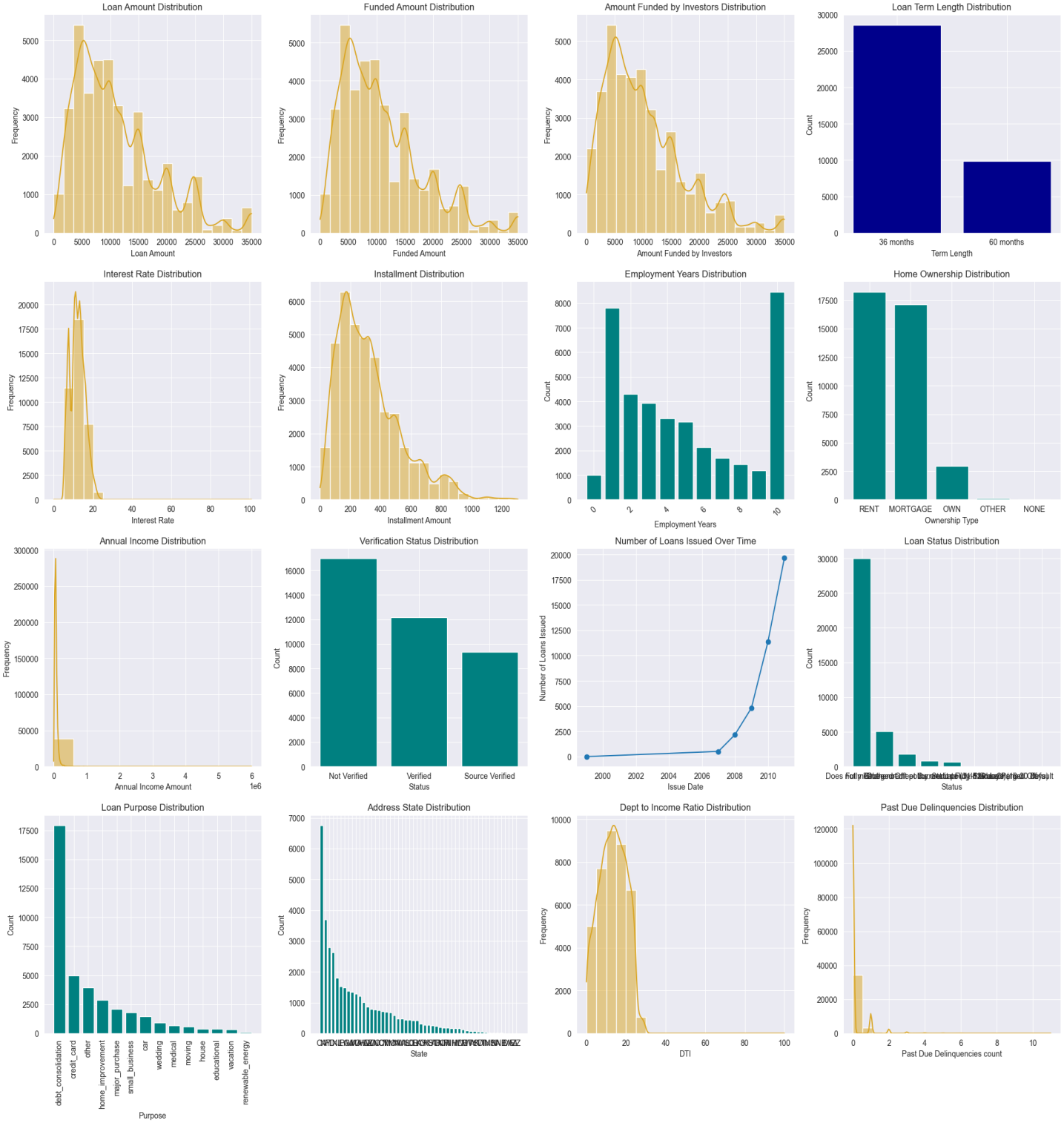
Exploratory Data Analysis:

We started by loading our data set and printing a statistical summary of the features. The data contained 36 features and over 38 thousand records.

Feature Description

1. **id**: This column appears to be an identifier for each entry in the dataset. It likely serves as a unique identifier for each record.
2. **member_id**: Another identifier, possibly representing a unique identifier for members associated with each loan.
3. **loan_amnt**: This column represents the amount of the loan requested by the borrower.
4. **funded_amnt**: The actual amount of the loan funded by the lender to the borrower.
5. **funded_amnt_inv**: The amount of the loan funded by investors.
6. **term**: The term length of the loan, usually in months.
7. **int_rate**: The interest rate on the loan, represented as a percentage.
8. **installment**: The monthly payment owed by the borrower.
9. **emp_length**: The length of time the borrower has been employed.
10. **home_ownership**: The type of home ownership (e.g., RENT, OWN, MORTGAGE).
11. **annual_inc**: The borrower's annual income.
12. **verification_status**: Indicates whether the income of the borrower was verified by the platform.
13. **issue_d**: The date the loan was issued.
14. **loan_status**: The current status of the loan (e.g., Current, Fully Paid, Charged Off).
15. **purpose**: The purpose of the loan as stated by the borrower.
16. **title**: A title provided by the borrower for the loan request.
17. **zip_code**: The first three digits of the borrower's zip code.
18. **addr_state**: The state of the borrower's address.
19. **dti**: Debt-to-Income ratio, representing the borrower's total monthly debt payments divided by their gross monthly income.
20. **delinq_2yrs**: The number of 30+ days past due delinquencies in the borrower's credit file.
21. **earliest_cr_line**: The date the borrower's earliest reported credit line was opened.
22. **inq_last_6mths**: The number of inquiries by creditors in the last 6 months.
23. **open_acc**: The number of open credit lines in the borrower's credit file.
24. **pub_rec**: The number of derogatory public records (bankruptcy filings, tax liens, or judgments) on the borrower's credit report.
25. **revol_bal**: Total credit-revolving balance (balance unpaid at the end of the credit card billing cycle).
26. **revol_util**: Revolving line utilization rate, or the amount of credit the borrower is using relative to their total available revolving credit.
27. **total_acc**: The total number of credit lines currently in the borrower's credit file.
28. **total_pymnt**: Total amount paid to date.
29. **total_pymnt_inv**: Total amount paid to date by investors.
30. **total_rec_prncp**: Total principal received to date.
31. **total_rec_int**: Total interest received to date.
32. **last_pymnt_d**: Last month payment was received.
33. **last_pymnt_amnt**: Last total payment amount received.
34. **next_pymnt_d**: Next scheduled payment date.
35. **last_credit_pull_d**: The most recent month LC pulled credit for this loan.
36. **repay_fail**: A binary indicator (0 or 1) indicating if the loan repayment failed.

We converted any columns with date-based values to date time format for easier processing. We used various charts and plots to visualize important features.

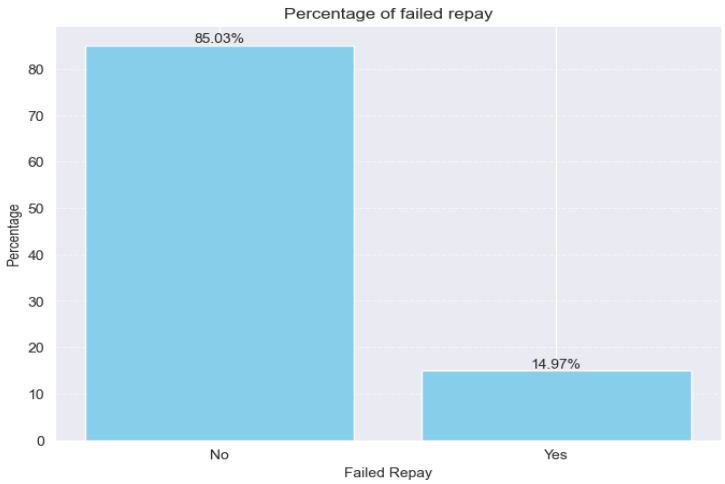


We proceeded to check the data for missing values. Two columns were found to have over 50% of their values missing. We chose to drop those columns. Then we dropped any remaining rows with missing values.

We removed any duplicate entries. We counted the number of unique values for each categorical column. Then, we used one-hot encoding to encode features with up to 50 unique values. We dropped any columns with unique-value count over 50.

Artificial Neural Network:

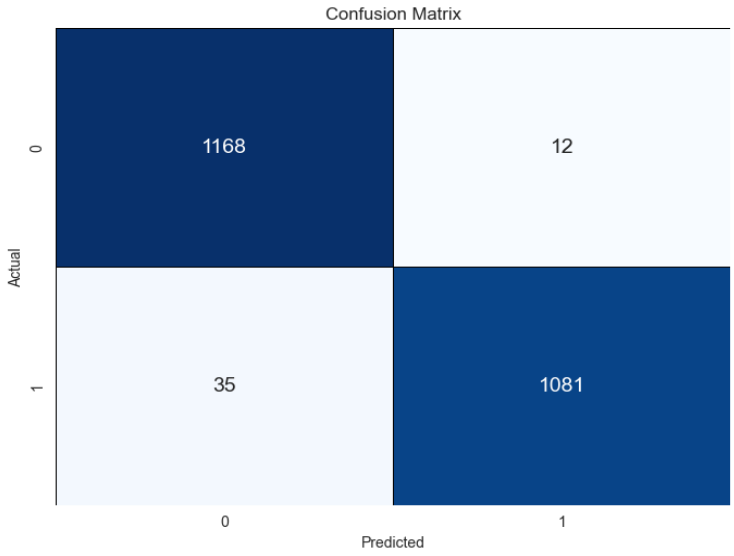
We started by checking for imbalance in class distribution in the target variable. We downsampled the majority class to match the minority class.



After the classes were balanced, we moved on to feature scaling. We used min-max scaling technique to scale our features. We splitted the data set to train set and test set. We converted our data to tensor format as recommended when working with tensorflow library. We broke down our data to random batches to cross-validate the training process. We defined the architecture of our network. The network consists of an input layer, four hidden layers, and an output layer. We used RELU activation for the hidden layers and Sigmoid activation for the output layer. We used a mix of rigid and lasso regularization with two hyperparameters L1 and L2. We chose the values for our hyperparameters by running our model over a range of different values for L1 and L2, and we settled on the values thar yielded the highest performance. We used the Adam optimizer for our network. As for the loss function, we used the binary cross entropy as recommended when working with sigmoid activation. We used three metrics to Evaluate our model accuracy, precision, and recall. We fitted our model to the data over 10 epochs.

```
Epoch 6/10
287/287 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9672 - loss: 0.3145 - precision_1: 0.9898 - recall_1: 0.9442 -
val_accuracy: 0.9739 - val_loss: 0.2900 - val_precision_1: 0.9981 - val_recall_1: 0.9480
Epoch 7/10
287/287 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9716 - loss: 0.2973 - precision_1: 0.9932 - recall_1: 0.9505 -
val_accuracy: 0.9782 - val_loss: 0.2952 - val_precision_1: 0.9734 - val_recall_1: 0.9821
Epoch 8/10
287/287 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9713 - loss: 0.2886 - precision_1: 0.9917 - recall_1: 0.9514 -
val_accuracy: 0.9787 - val_loss: 0.2623 - val_precision_1: 0.9917 - val_recall_1: 0.9642
Epoch 9/10
287/287 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9673 - loss: 0.2769 - precision_1: 0.9860 - recall_1: 0.9478 -
val_accuracy: 0.9804 - val_loss: 0.2549 - val_precision_1: 0.9945 - val_recall_1: 0.9651
Epoch 10/10
287/287 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9720 - loss: 0.2656 - precision_1: 0.9911 - recall_1: 0.9535 -
val_accuracy: 0.9795 - val_loss: 0.2458 - val_precision_1: 0.9890 - val_recall_1: 0.9686
72/72 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
```

We set a threshold of 0.5 for our model. Finally, we used a confusion matrix to visualize the results and printed a classification report.




```
Classification Report:
              precision    recall  f1-score   support

         0.0       0.97      0.99      0.98      1180
         1.0       0.99      0.97      0.98      1116

    accuracy                           0.98      2296
   macro avg       0.98      0.98      0.98      2296
weighted avg       0.98      0.98      0.98      2296
```

Also worth mentioning that throughout the entire project we relied on version control systems and platforms like Git and GitHub to organize our work and keep track of tasks and issues that needed to be addressed.

CREDITS:

Fares Mohamed Salah **ID:** 22011614

Kareem Ashraf Ahmed **ID:** 22060171

Abdelrahman Ahmed El-Motawakel **ID:** 22010456

Our project repo can be found at: https://github.com/faresmohamed260/School-Work/tree/main/Neural%20Networks/Project_2

THANKS 😊