Team 3

Bank Loan Case Study

Dinan Sooriyaarachchi, Louisa Anjanette Auwlia, Benjamin Tran, Brandon Tang, Amelie Carrillo, Keira Sakamoto

Overview

01. Data Cleaning

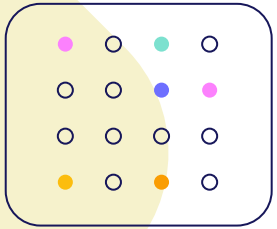
02. Exploratory Analysis

03. Logistic Regression Model

04. Cost Benefit Analysis

05. Interpretation, Recommendation

01. Data Cleaning



Data Cleaning Timeline

Dropped Irrelevant Columns: Removed columns that were not necessary for the analysis, such as LoanNr_ChkDgt, Name, Bank, City, Zip, CreateJob, and RetainedJob.

Dropped Duplicate Rows: Identified and removed duplicate rows to maintain data uniqueness.

Dropped Rows with Missing Values: Removed rows with missing values in critical columns like State, BankState, NewExist, RevLineCr, LowDoc, DisbursementDate, and MIS_Status.

Removed Foreign Values: Removed values in columns that had values that didn't match with what was supposed to be there (e.g. removing values like 'a' from column where valid values are only 0 and 1)

Applied Binary Encoding to Categorical Columns: Converted RevLineCr, LowDoc, and UrbanRural into binary (1 or 0) values for machine learning compatibility.
Output: One-hot encoded categorical columns.

Applied One-Hot Encoding to NAICS Column: Transformed NAICS into one-hot encoded columns to represent industry categories numerically.

Data Cleaning

```
# Dropping irrelevant columns
cols_to_drop = ['LoanNr_ChkDgt', 'Name', 'Bank', 'City', 'Zip', 'CreateJob', 'RetainedJob']

df.drop(cols_to_drop, axis=1, inplace=True)

print('Dropped columns: ', cols_to_drop)
```

```
# Drop duplicate rows
df = df.drop_duplicates()
print('Duplicate rows have been dropped')
```

```
# Drop all rows with missing fields

missing_data = ['State', 'BankState', 'NewExist', 'RevLineCr', 'LowDoc', 'DisbursementDate', 'MIS_Status']

df.dropna(subset=missing_data, inplace=True)

print('Dataframe Shape after dropping: ', df.shape)
```

Data Cleaning

```
# Dropping foreign values and binary encoding
# LowDoc
df.dropna(subset=['LowDoc'], inplace=True)
df = df[df['LowDoc'].isin(['N', 'Y', '1', '0'])].copy()
df['LowDoc'] = df['LowDoc'].map({'Y': '1','N': '0','1': '1','0': '0'})
df['LowDoc'] = df['LowDoc'].astype(int)
```

```
# RevLineCr
df.dropna(subset=['RevLineCr'], inplace=True)
df = df[df['RevLineCr'].isin(['N', 'Y', '1', '0'])].copy()
df['RevLineCr'] = df['RevLineCr'].map({'Y': '1','N': '0','1': '1','0': '0'})
df['RevLineCr'] = df['RevLineCr'].astype(int)
```

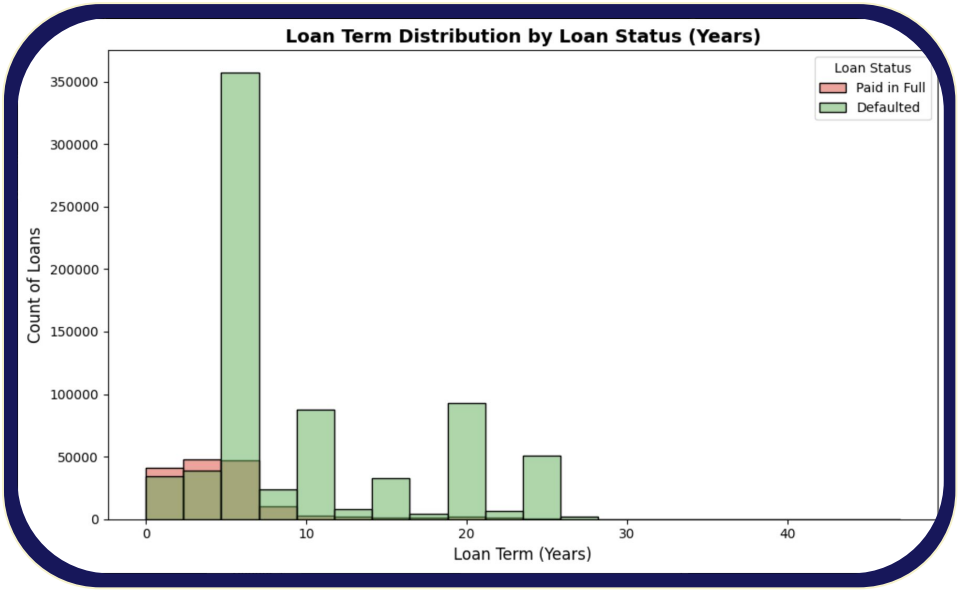
```
# UrbanRural
df = df[df['UrbanRural'].isin([1, 2])]

# Convert 'UrbanRural' to integer
df['UrbanRural'] = df['UrbanRural'].astype(int)

# Map 1 -> 0 and 2 -> 1
df['UrbanRural'] = df['UrbanRural'].map({1: 0, 2: 1})
```

In [9]:
```
# Define NAICS code to category mapping
naics_mapping = {
    "0" : None,
    "11": "Agriculture_Forestry_Fishing_Hunting",
    "21": "Mining_Quarrying_Oil_GasExtraction",
    "22": "Utilities",
    "23": "Construction",
    "31-33": "Manufacturing",
    "42": "WholesaleTrade",
    "44-45": "RetailTrade",
    "48-49": "Transportation_Warehousing",
    "51": "Information",
    "52": "Finance_Insurance",
    "53": "RealEstate_Rental_Leasing",
    "54": "Professional_Scientific_TechnicalServices",
    "55": "ManagementOfCompanies_Enterprises",
    "56": "Administrative_Support_WasteManagement_RemediationServices",
    "61": "EducationalServices",
    "62": "HealthCare_SocialAssistance",
    "71": "Arts_Entertainment_Recreation",
    "72": "Accommodation_FoodServices",
    "81": "OtherServices",
    "92": "PublicAdministration"
}

# Extract the first two digits
df['NAICS'] = df['NAICS'].astype(str).str[:2]

# Map the two-digit code to a category
df['NAICS'] = df['NAICS'].map(naics_mapping)

# Drop the indexes where NAICS is None or NaN
df = df.dropna(subset=['NAICS'])
```

02. Exploratory Analysis

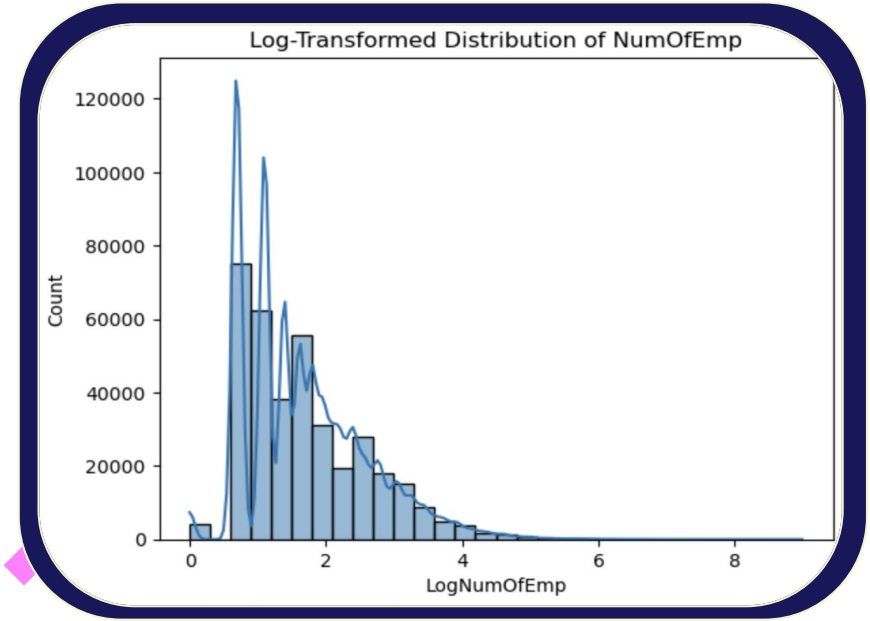
Visualizations

By creating the "Term_Years" column which was derived from the "Term" column, we converted loan terms from months to years for easier interpretation. This histogram shows how loan terms relate to loan status (Paid in Full vs. Defaulted). This reveals trends like shorter-term loans being more likely to be repaid, while longer terms might show higher default rates.



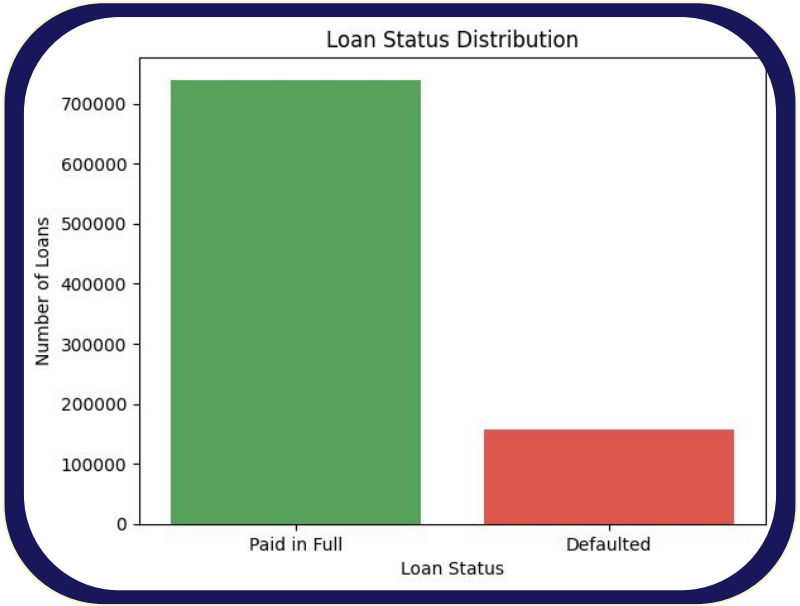
Visualization

By log-transforming the "NumOfEmp" we are able to see that the majority of businesses have a small number of employees, as indicated by the high frequency near lower log values, while fewer businesses have larger workforces. The log transformation smooths the data, making the distribution easier to interpret.



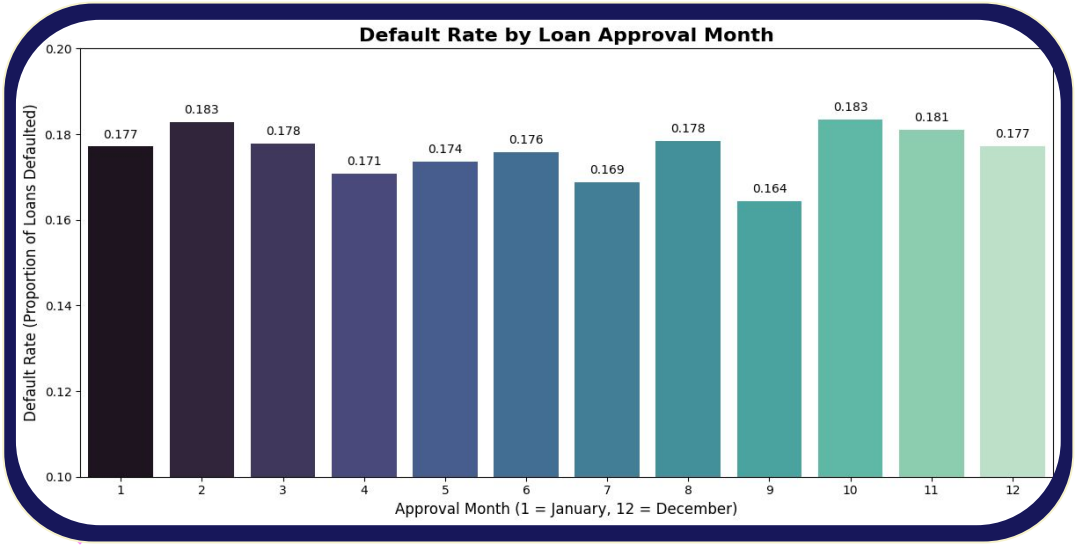
Visualization

Shows the distribution of loan statuses. The green bar represents loans that were ”Paid in Full,” which are significantly higher in number compared to the red bar, representing ”Defaulted” loans. This imbalance highlights the need to address the class imbalance in the dataset, as it could impact the performance of the predictive model.



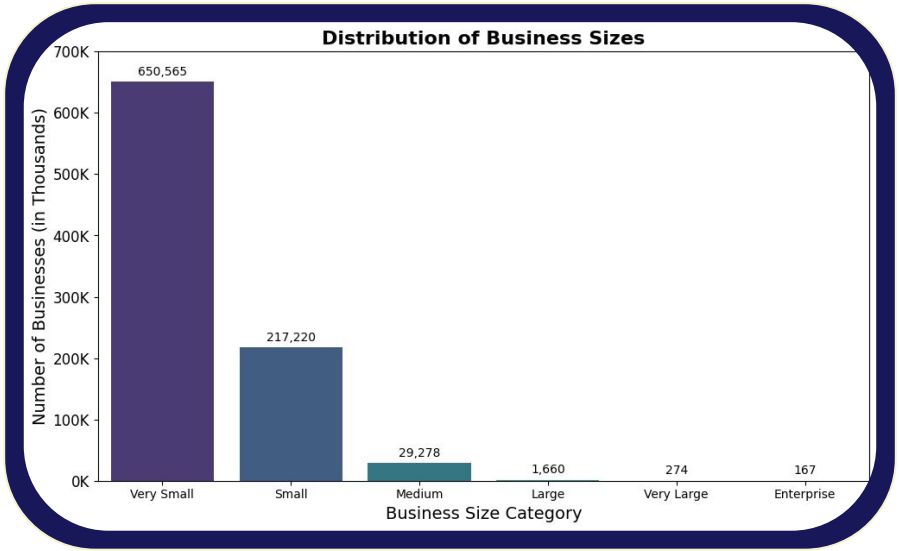
Visualization

The graph shows default rates across loan approval months, with the lowest in September (0.164) and the highest in February and October (0.183). Most months have similar default rates around 0.176–0.183, indicating consistent patterns except for seasonal variations. Lenders should investigate higher default months to adjust approval strategies, while borrowers perform better with loans approved in September.



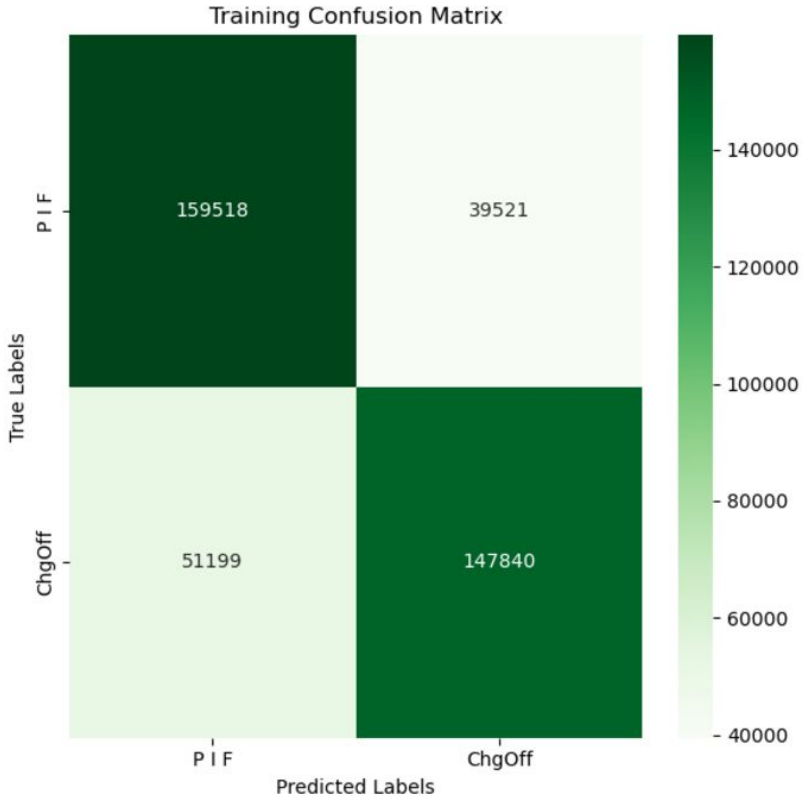
Visualization

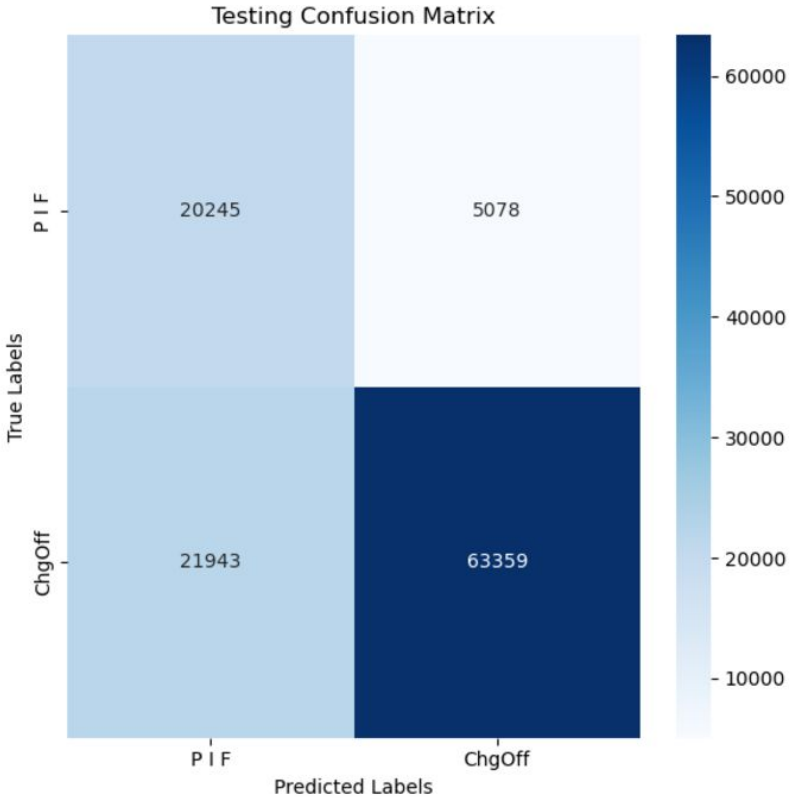
The graph depicts the distribution of businesses by size categories. “Very Small” businesses dominate the dataset, comprising 650,565 entities, followed by ”Small” businesses at 217,220. In contrast, larger categories such as ”Medium” (29,278), ”Large” (1,660), ”Very Large” (274), and ”Enterprise” (167) have significantly fewer businesses, emphasizing the predominance of smaller enterprises in the dataset.



03. Logistic Regression Model

Confusion Matrix





Classification Report

Training Classification Report:

	precision	recall	f1-score	support
0	0.76	0.80	0.78	199039
1	0.79	0.74	0.77	199039
accuracy			0.77	398078
macro avg	0.77	0.77	0.77	398078
weighted avg	0.77	0.77	0.77	398078

Testing Classification Report:

	precision	recall	f1-score	support
0	0.48	0.80	0.60	25323
1	0.93	0.74	0.82	85302
accuracy			0.76	110625
macro avg	0.70	0.77	0.71	110625
weighted avg	0.82	0.76	0.77	110625

Specificity

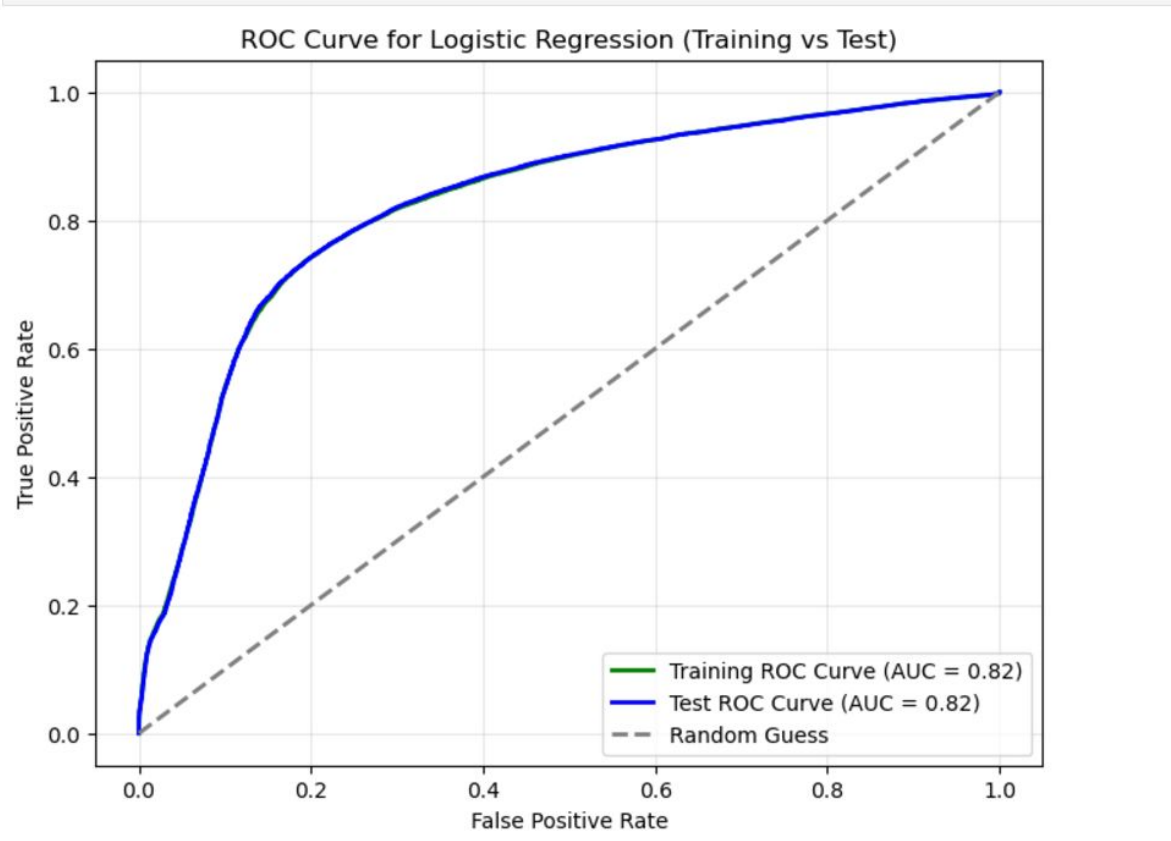
```
y_pred = logit_reg.predict(X_test)

from sklearn.metrics import confusion_matrix

# Assuming y_test (true labels) and y_pred (predicted labels) are available
cm = confusion_matrix(y_test, y_pred)

# Extract values from confusion matrix
TN = cm[0, 0]  # Top-left value: True  Negatives
FP = cm[0, 1]  # Top-right value: False Positives

# Calculate Specificity
specificity = TN / (TN + FP)

print(f"Specificity: {specificity:.2f}")
```

```
Specificity: 0.80
```

$$\text{Specificity} = \frac{\text{True Negatives (TN)}}{\text{True Negatives (TN)} + \text{False Positives (FP)}}$$

- **True Negatives (TN):** Loans predicted as "No Default/PIF" that truly didn't default.
- **False Positives (FP):** Loans predicted as "No Default/PIF" but actually defaulted.

ROC Curve


ROC Curve for Logistic Regression (Training vs Test)
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Training ROC Curve (AUC = 0.82)
Test ROC Curve (AUC = 0.82)
Random Guess

04. Cost Benefit Analysis

Cost-Benefit Analysis

Benefits

Reduction in Risk

- High precision for defaults (*0.93*) minimizes the risk of approving high-risk loans

- By identifying *74%* of true defaults (recall), the model provides reliable insights to reduce defaulted loan rates

Improved Lending Strategy

- Correctly predicting defaults effectively allows lenders to tailor loan terms, request additional collateral, or deny high-risk applications outright, ensuring profitability and reduced write-offs

Costs

False Positives *5078 (4%) loans misclassified as defaults*

- May lead to unnecessarily stringent loan terms or rejections for loans that would have been profitable

- Opportunity cost of losing potential revenue from low-risk borrowers

False Negatives *21,943 (20%) loans misclassified as non-defaults*

- These loans pose significant financial risk, as they represent defaults the model failed to identify

- Could lead to increased loan loss reserves and harm the institution's financial stability

Imbalanced Precision

- Lower precision for non-defaults suggests that improving the model's ability to differentiate non-defaulted loans is critical to avoiding unnecessary costs from misclassified loans

Cost Benefit Analysis

```python
import numpy as np
import pandas as pd
from sklearn.metrics import confusion_matrix

def calculate_expected_profit(y_true, y_pred, disburse_gross, benefit_ratio=0.05, cost_ratio=0.25):
    # Ensure inputs are numpy arrays
    y_true = np.array(y_true)
    y_pred = np.array(y_pred)
    disburse_gross = np.array(disburse_gross)

    # Validate inputs length
    if not (len(y_true) == len(y_pred) == len(disburse_gross)):
        raise ValueError("Length of y_true, y_pred, and disburse_gross must match")

    # Identify True Positives and False Positives
    true_positive = (y_true == 1) & (y_pred == 1)
    false_positive = (y_true == 0) & (y_pred == 1)

    # Calculate profit/loss for each row
    profit_per_row = np.where(true_positive, disburse_gross * benefit_ratio, 0) - \
                     np.where(false_positive, disburse_gross * cost_ratio, 0)

    # Calculate the average profit
    avg_profit = profit_per_row.mean()

    return avg_profit

np.random.seed(42)  # For reproducibility

# Generate random disbursement values for the test dataset
disbursement_gross_test = np.random.uniform(5000, 50000, size=len(y_test))

# Define benefit and cost ratios
benefit_ratio = 0.05  # 5% profit for True Positives
cost_ratio = 0.25     # 25% loss for False Positives

# Calculate the average expected profit
avg_profit = calculate_expected_profit(y_test, y_test_pred, disbursement_gross_test, benefit_ratio, cost_ratio)

print(f"Average Expected Profit: ${avg_profit:.2f}")
```

```
Average Expected Profit: $461.34
```

Interpretations

- Our model prioritizes minimizing false positives over minimizing false negatives because it is more costly to give out a loan to a company that is going to default versus not giving out a loan to a company that is going to pay the loan back fully
- Our cost benefit analysis shows us that the bank makes an average of $461 per loan, which seems low, but makes sense because the risk of losing is 5 times as bad as correctly classifying a loan
- In a real life scenario, bank employees will use this model to decide whether to give out a loan or not.
- As time passes, more data points can be added to the dataset periodically to keep the model updated and to see if changes need to be made to improve the model

Thanks!