Indian Institute of Technology Hyderabad

Fraud Analytics (CS6890)

Assignment: 3 | Example-dependent cost-sensitive regression

Name	Roll Number
Shreesh Gupta	CS23MTECH12009
Hrishikesh Hemke	CS23MTECH14003
Manan Patel	CS23MTECH14006
Yug Patel	CS23MTECH14019
Bhargav Patel	CS23MTECH11026

Contents

1 | Problem statement

- The objective of this assignment is to develop a cost-sensitive regression model that accounts for varying costs associated with different types of errors in predictions, specifically false negatives, true positives, and false positives.
- Approaches:
  1. **Bahnsen's Approach:**
     - Bahnsen's approach focuses on adjusting the cost matrix to reflect the asymmetrical costs associated with different types of prediction errors.
     - The cost matrix is modified to include the false negative cost (column M), true positive cost (constant at 6), false positive cost (constant at 6), and true negative cost (constant at 0).
     - The model is trained using this adjusted cost matrix to optimize predictions while considering the cost-sensitive nature of the problem.
  2. **Nikou Gunnemann's Approach:**
     - Nikou Gunnemann's approach incorporates cost-sensitive learning by directly incorporating the cost matrix into the model's loss function.
     - The cost matrix is defined based on the false negative, true positive, false positive, and true negative costs as described above.
     - The model is trained to minimize the expected cost, taking into account the different costs associated with prediction errors.

2 | Description of the data set

We have one csv file, "costsensitiveregression.csv".The dataset contains 147,636 entries with 13 columns.

- The dataset contains several columns (A to K) representing independent variables related to payment transactions.
- Column L represents the dependent variable, which could be, for example, the risk level associated with each transaction.
- Column M contains the false negative cost, which varies for each transaction based on risk parameters.

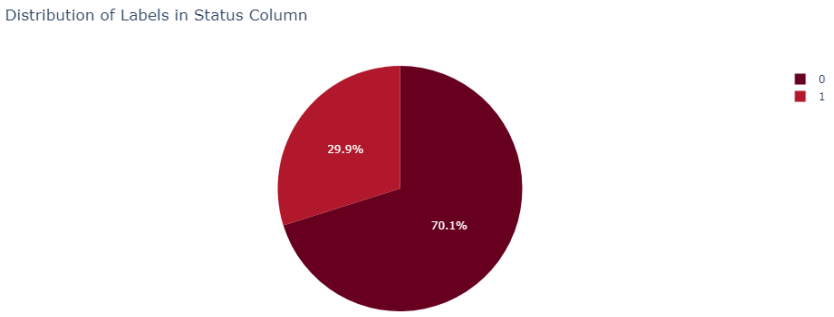

Figure 2.1: Dataset visualisation

- **NotCount:** An integer count,indicating the number of certain events or occurrences where the outcome was "No."
- **YesCount**: Similarly, this is an integer count for outcomes that were "Yes."
- ATPM: Appears to be a numeric feature, possibly a rate or probability (mean close to 0.25)

3 | Algorithm Used

3.1 | Bahnsen Approach

3.1.1 | Introduction

This implementation details a cost-sensitive logistic regression model that optimizes its parameters using gradient descent. Unlike traditional logistic regression, this model considers the varying costs of different types of classification errors, which can be crucial in domains like healthcare, finance, or any field where the consequences of errors are asymmetric.

3.1.2 | Model Description

Sigmoid Function

The logistic regression model uses the sigmoid function to map the net input (z) (a linear combination of input features (X) and weights (w)) to probabilities between 0 and 1. The sigmoid function is defined as:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Cost-Sensitive Loss Function

The cost-sensitive loss function incorporates costs associated with four types of outcomes: true positives, true negatives, false positives, and false negatives. It is defined as follows:

$$L(y, \hat{y}, C) = \sum_{i=1}^{N} [y_i \times ((1 - \hat{y}_i) \times CFN + \hat{y}_i \times CTP) + (1 - y_i) \times (\hat{y}_i \times CFP + (1 - \hat{y}_i) \times CTN)]$$

Here:

- y_i is the actual label of the ith sample.
- $\hat{y}_i$ is the predicted probability for the ith sample.
- C_{TP}, C_{FN}, C_{FP}, C_{TN} are the costs associated with true positives, false negatives, false positives, and true negatives, respectively.

3.1.3 | Gradient Descent Optimization

The model parameters (weights (w) and bias (b)) are updated using gradient descent to minimize the cost-sensitive loss. The gradients for the weights and bias are computed as:

$$\frac{\partial L}{\partial w} = -\frac{1}{N} \sum_{i=1}^{N} X^T \times ((y - \hat{y}) \times (CTP - CFP))$$

$$\frac{\partial L}{\partial b} = -\frac{1}{N} \sum_{i=1}^{N} ((y - \hat{y}) \times (CTP - CFP))$$

The weights and bias are then updated iteratively:

$$w = w - \alpha \times \frac{\partial L}{\partial w}$$

$$b = b - \alpha \times \frac{\partial L}{\partial b}$$

where α is the learning rate.

Algorithm 1 Cost-Sensitive Logistic Regression using Gradient Descent (Bahnsen approach)

Input: Training data X, labels y, cost matrix C, learning rate α, number of epochs *epochs*
Initialize weights w and bias b to zeros
for i in *epochs* **do**
  Compute probabilities: $probabilities \leftarrow \text{sigmoid}(Xw + b)$
  Compute gradients:
    $dw \leftarrow -\frac{1}{N} X^T((y - probabilities) \times (C_{TP} - C_{FP}))$
    $db \leftarrow -\frac{1}{N} \sum((y - probabilities) \times (C_{TP} - C_{FP}))$
  Update weights and bias:
    $w \leftarrow w - \alpha \times dw$
    $b \leftarrow b - \alpha \times db$
end for
Output: Trained weights w and bias b =0

3.1.4 | Results

Metric	Value
Accuracy	0.2987
Log Loss	0.6931
Generations	50

Table 3.1: Results Obtained Using Bahnsen Approach

3.2 | Nikou Gunnemann's Approach

The `logistic_loss` function in Python computes a variant of the logistic regression loss, incorporating costs that vary based on different model variants. This adaptation allows the model to differently weigh the importance of misclassifications according to their associated costs, crucial in scenarios where different types of errors have significantly different implications.

3.2.1 | General Setup for Logistic Regression

Logistic regression predicts the probability (p) that a given instance (x) belongs to the positive class (i.e., ($y = 1$)):

$$p = \sigma(\beta^T x)$$

where (σ) is the sigmoid activation function defined by:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

The conventional logistic loss function, used for training logistic regression models, penalizes predictions based on the log likelihood of the true classes:

$$L(y, p) = -[y \log(p) + (1 - y) \log(1 - p)]$$

This function is summed over all instances, where being wrong and confident leads to higher penalties.

3.2.2 | Nikou Gunnemann's Approach for Cost-Sensitive Regression

Nikou Gunnemann's approach for cost-sensitive regression extends the conventional logistic loss function to incorporate costs associated with different types of misclassifications. This cost-sensitive variant of logistic regression is particularly useful in scenarios where the consequences of different types of errors vary significantly.

The modified cost-sensitive logistic loss function can be expressed as:

$$L_{\text{CS}}(y, p, C) = -[y \cdot \log(p) \cdot C_{TP} + (1 - y) \cdot \log(1 - p) \cdot C_{TN}]$$
$$-[y \cdot \log(1 - p) \cdot C_{FP} + (1 - y) \cdot \log(p) \cdot C_{FN}]$$

where:

- C_{TP}, C_{TN}, C_{FP}, C_{FN} are the costs associated with true positives, true negatives, false positives, and false negatives, respectively.

This cost-sensitive logistic loss function allows the model to learn from misclassifications differently based on the costs assigned to each type of error. It provides a flexible framework to handle scenarios where the importance of different types of errors varies significantly, ensuring the model's predictions align with the specific requirements of the problem domain.

3.3 | Variant-Specific Cost-Sensitive Loss Functions

3.3.1 | Variant A: Linear Weighting

In Variant A, the weights (a_i) are directly proportional to the costs associated with each instance, given as $a_i = c_i$. The exponents (b_i) are set to 1, maintaining the standard logistic loss form but scaled by cost.

The cost-sensitive logistic loss function for Variant A can be expressed as:

$$L_A = \sum_i c_i [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)]$$

This variant emphasizes instances with higher costs, leading to increased penalties for misclassifications associated with higher-cost instances.

3.3.2 | Variant B: Exponential Weighting

In Variant B, the weights (a_i) are all set to 1, while the exponents (b_i) are determined by the inverse gamma function of the costs, modifying the sensitivity of the loss to errors. The exponents are calculated as $b_i = \Gamma^{-1}(c_i) - 1$.

The cost-sensitive logistic loss function for Variant B can be expressed as:

$$L_B = \sum_i [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)]^{b_i}$$

This variant adjusts the loss sensitivity based on the costs, with higher-cost instances influencing the loss function more significantly.

3.3.3 | Variant C: Ratio Control

In Variant C, both the weights (a_i) and exponents (b_i) are tailored to manage the ratio of costs between different types of misclassifications. Calculations involve solving equations to balance the ratios appropriately.

The cost-sensitive logistic loss function for Variant C can be expressed as:

$$L_C = \sum_i \frac{1}{\Gamma(b_i + 1)} [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)]^{b_i}$$

This variant ensures that the loss function is adjusted to maintain specific cost-benefit ratios between different types of misclassifications.

3.3.4 | Variant D: Maintaining Constant Correct Classification Cost

In Variant D, the weights (a_i) and exponents (b_i) are designed to keep the cost of correct classifications constant while varying the penalties for misclassifications according to their costs. The adjustments are made to ensure that the losses for correct classifications remain constant across different costs.

The cost-sensitive logistic loss function for Variant D can be expressed as:

$$L_D = \sum_i \frac{0.5}{\Gamma(b_i + 1) - \Gamma(b_i + 1, 0.6931)} [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)]^{b_i}$$

This variant maintains a balanced approach where correct classifications are consistently penalized regardless of the associated costs, while misclassifications are adjusted based on their cost implications.

These variants modify the standard logistic loss to integrate the cost directly into the model training, making the logistic regression sensitive to the cost associated with incorrect predictions, which is especially useful in uneven cost scenarios.

4 | Implementation of Nikou Gunnemann's Approach

Algorithm 2 Customized Logistic Loss Computation

Input: X (feature matrix), y (target vector), *costs* (cost vector), *variant* (variant type)
Output: Customized logistic loss value
for each β in parameter range **do**
  Compute predictions: $predictions \leftarrow \text{sigmoid}(\mathbf{X} \cdot \beta)$
  Initialize loss: $loss \leftarrow 0$
  for i from 1 to m **do**
    if variant = 'A' **then**
      Compute weights: $a_i \leftarrow \text{costs}_i, b_i \leftarrow 1$
    else if variant = 'B' **then**
      Compute weights: $a_i \leftarrow 1, b_i \leftarrow \text{gammaincinv}(\text{costs}_i, 0.5) - 1$
    else if variant = 'C' **then**
      Compute weights: $b_i \leftarrow \text{gammaincinv}(1, (1 + 0.5 \cdot \text{costs}_i)/(\text{costs}_i + 1)), a_i \leftarrow 1/\text{gamma}(b_i + 1)$
    else if variant = 'D' **then**
      Compute weights: $b_i \leftarrow \text{gammaincinv}(1, (1 + 0.5 \cdot \text{costs}_i)/(\text{costs}_i + 1)), a_i \leftarrow 0.5/(\text{gamma}(b_i + 1) - \text{gammainc}(b_i + 1, 0.6931))$
    end if
    Update loss: $loss \leftarrow loss + a_i \cdot (y_i \cdot (-\log(predictions_i + 1e-15))^{b_i} + (1 - y_i) \cdot (-\log(1 - predictions_i + 1e-15))^{b_i})$
  end for
  Store computed loss
end for
Return minimum loss and corresponding parameters =0

4.1 | Fitting Cost-Sensitive Logistic Regression Models

The `fit_model` function is designed to fit logistic regression models by optimizing a custom logistic loss function that takes into account different cost structures depending on the specified variant. The function employs the `minimize` method from `scipy.optimize` to find the best model coefficients (β) that minimize the loss.

4.1.1 | Function Definition

The `fit_model` function takes the following parameters:

- X (array-like): Feature matrix for the training data.
- y (array-like): Target vector for the training data.
- `costs` (array-like): Vector of costs associated with each instance in the training data.
- `variant` (str): Specifies the variant of the cost-sensitive logistic regression to be used.

It initializes the model coefficients to zeros, then uses the BFGS algorithm (a quasi-Newton method) to optimize the coefficients by minimizing the custom logistic loss function. The BFGS algorithm is a gradient-based optimization technique that iteratively adjusts the coefficients to reduce the loss until convergence.

4.1.2 | Optimization Process:

The optimization process involves iteratively updating the model coefficients (β) using the BFGS algorithm to minimize the custom logistic loss function. The algorithm computes the gradient of the loss function with respect to the coefficients and adjusts them in the direction that reduces the loss.

Upon convergence, the `fit_model` function returns the optimized model coefficients (β) that best fit the cost-sensitive logistic regression model to the training data.

Variant	Coefficient 1	Coefficient 2	Coefficient 3	Coefficient 4
A	0.1190579	-0.15401323	0.05821664	0.21904436
B	-0.00328367	0.00659971	-0.28634056	0.02109848
C	0.167472449	-0.146788075	-1.45220745	0.017632433
D	0.169746606	-0.153645542	-1.35997630	0.022342908

Table 4.1: Model Coefficients for Different Variants

4.2 | Logistic Regression Overview and Performance Evaluation

4.2.1 | Logistic Regression Explanation

Logistic Regression is a statistical method used for binary classification that predicts the probability of the target variable's categories. It models the probability of the default class (usually "1") using the logistic function.

4.2.2 | Equation of the Logistic Regression Model

The logistic regression model calculates probabilities using the logistic function:

$$\text{logit}(p) = \log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n$$

Where:

- p is the probability of belonging to the positive class.
- $\beta_0, \beta_1, ..., \beta_n$ are the coefficients of the model.
- $x_1, x_2, ..., x_n$ are the predictor variables.

4.2.3 | Probability Prediction Using the Sigmoid Function

The logistic function or the sigmoid function is used to convert the linear regression output to a probability:

$$p = \sigma(z) = \frac{1}{1+e^{-z}}$$

where z is the linear combination of the features and coefficients, $z = \beta_0 + \beta_1 x_1 + \ldots + \beta_n x_n$.

4.2.4 | Model Training and Performance

Training Process: A standard logistic regression model was trained using the `LogisticRegression` class from `sklearn`, configured with a maximum of 500 iterations to ensure convergence.

4.2.5 | Model Performance Evaluation

Performance of the trained logistic regression model was evaluated using metrics such as accuracy, precision, recall, F1 score, and confusion matrix. These metrics provide insights into the model's ability to correctly classify instances and its overall performance in binary classification tasks.

4.3 | Cost Comparison between Standard and Cost-Sensitive Logistic Regression Models

The bar chart illustrates a stark contrast in the total misclassification costs incurred by two logistic regression models: the standard LR and the cost-sensitive LR.

Standard Logistic Regression (Blue Bar): The cost incurred by the standard logistic regression model is substantially higher, with a total of 812,473.36. This suggests that while the model might predict accurately, it does not account for the varying costs of different types of classification errors, leading to a less economically optimal outcome.

Cost-Sensitive Logistic Regression (Green Bar): In contrast, the cost-sensitive logistic regression model shows a significantly reduced cost of 177,162.0. This model incorporates a cost matrix that assigns specific costs to different types of errors, such as false positives and false negatives. By doing so, it is able to minimize the more economically impactful errors, thus reducing the overall cost.

Figure 4.1: Frequency vs TrustRank score

5 | Results

5.1 | F1 Score Comparison Between Standard and Cost-Sensitive Logistic Regression

The F1 Score is the harmonic mean of precision and recall, providing a balance by accounting for both false positives and false negatives. It is particularly valuable in scenarios where class distributions are imbalanced.

5.1.1 | Mathematical Representation

The F1 Score is defined mathematically as:

$$F1 = 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

where:

- **Precision** is the proportion of true positive predictions to the total predicted positives:

$$\text{Precision} = \frac{TP}{TP + FP}$$

- **Recall** (also known as sensitivity) is the ratio of true positive predictions to the actual positives:

$$\text{Recall} = \frac{TP}{TP + FN}$$

Standard Logistic Regression

- **F1 Score**: 0.763, indicating a relatively high balance between precision and recall, suggesting that this model may predict positive classes more reliably than the cost-sensitive counterpart.

Cost-Sensitive Logistic Regression

- **F1 Score**: 0.460, significantly lower than the standard model, suggesting a trade-off between precision and recall that favors minimizing costlier errors.
- **Discussion** The higher F1 Score of the standard logistic regression model indicates better performance in terms of balanced precision and recall. However, this model does not account for the different costs of misclassification. In contrast, the cost-sensitive logistic regression model, despite its lower F1 Score, may be more effective in contexts where reducing costs associated with misclassification is critical. It potentially sacrifices precision and/or recall to reduce more costly errors, which aligns with its optimization objectives.

  The choice between these models should be based on specific context and objectives: whether minimizing overall misclassification costs is more critical than maintaining a balance between precision and recall.

Table 5.1: Comparison of F1 Scores for Logistic Regression Models

Model Description	F1 Score
Standard Logistic Regression	0.7636
Bahnsen Approach Cost-Sensitive LR	0.4600
Nikou Gunnemann's Approach Variant A	0.7564
Nikou Gunnemann's Approach Variant B	0.1850
Nikou Gunnemann's Approach Variant C	0.7664
Nikou Gunnemann's Approach Variant D	0.7668

Table 5.2: Comparison of Costs for Logistic Regression Models

Model Description	Cost (USD)
Standard Logistic Regression	$39,207,831,424.14
Bahnsen Approach Cost-Sensitive LR	$5,231,416,704.00
Nikou Gunnemann's Approach Variant A	$42,817,146,185.82
Nikou Gunnemann's Approach Variant B	$104,513,950,986.58
Nikou Gunnemann's Approach Variant C	$37,255,560,550.79
Nikou Gunnemann's Approach Variant D	$37,630,307,950.06

5.2 | Confusion Matrix Comparison

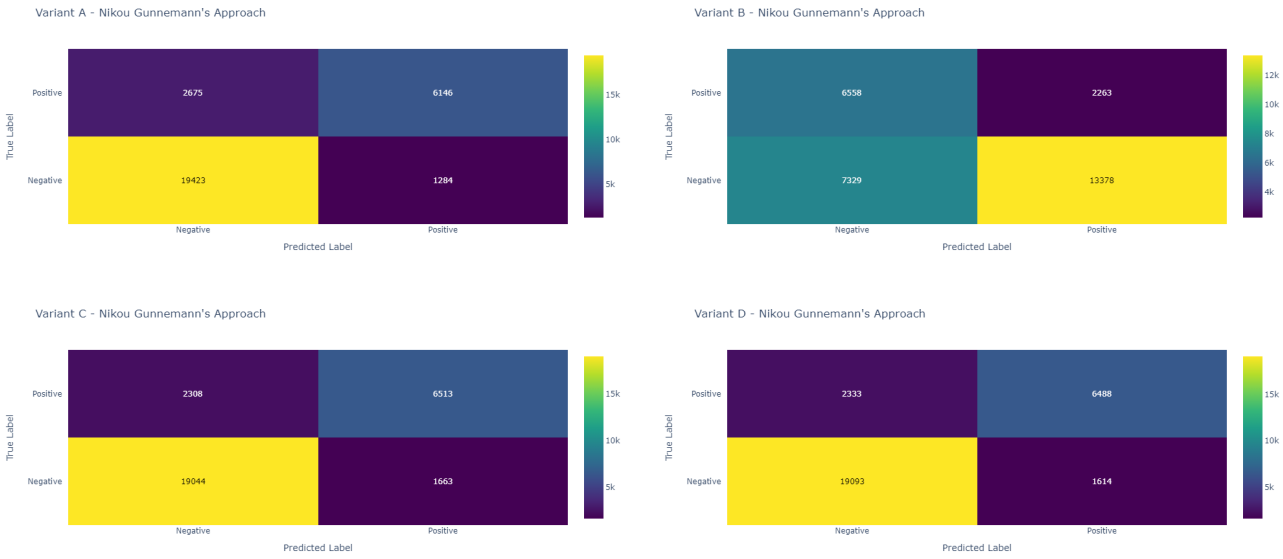

Figure 5.1: Confusion Matrix of different variants of Nikou Gunnemann's method

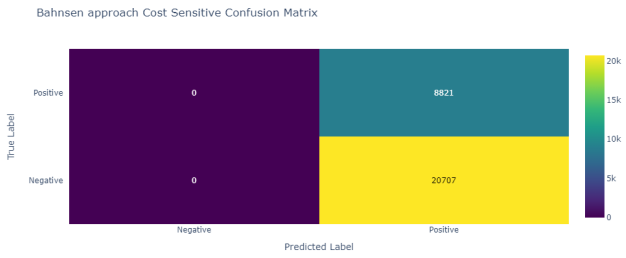

Figure 5.2: Bahnsen approach cost sensitive confusion matrix

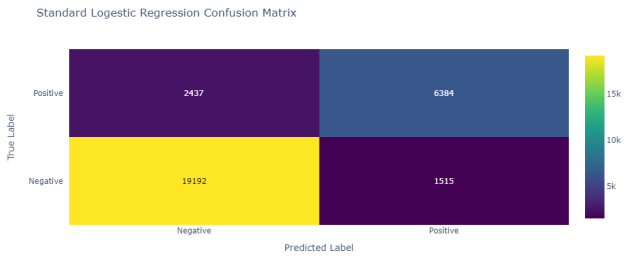

Figure 5.3: Standard logistic regression confusion matrix

5.3 | Cost Matrix Analysis for Logistic Regression Models

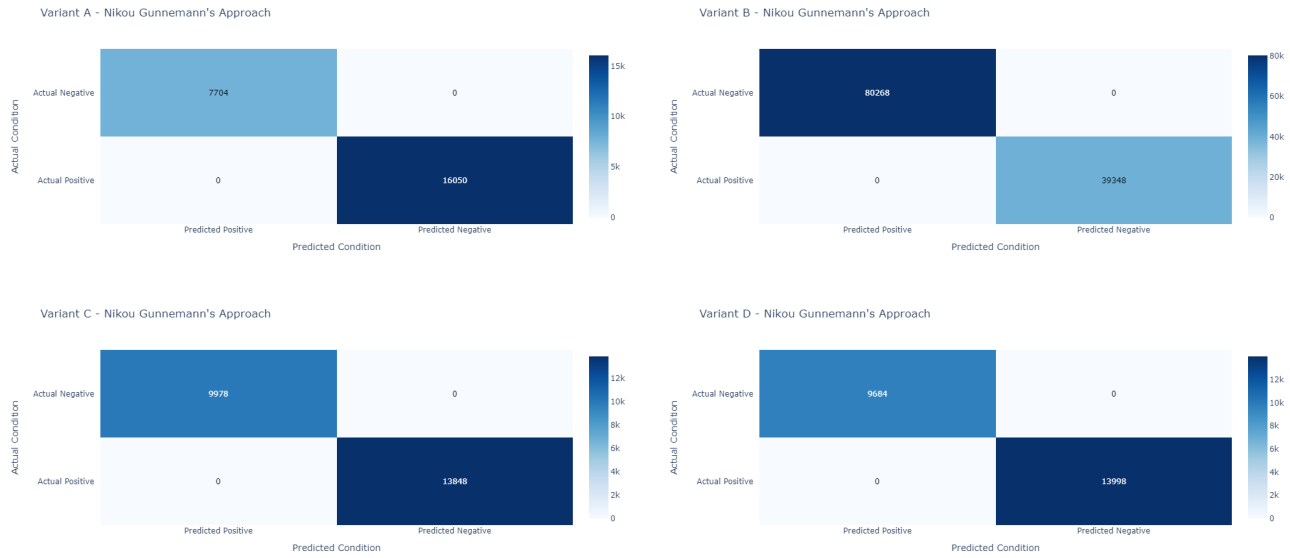

Figure 5.4: Cost Matrix analysis for different variants of Nikou Gunnemann's method

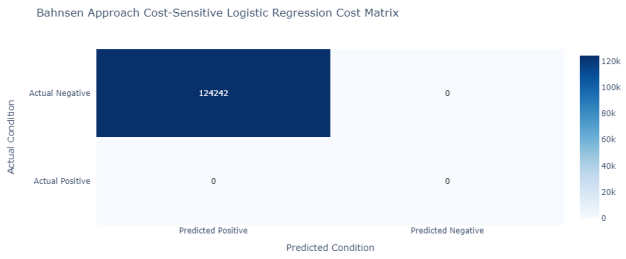

Figure 5.5: Bahnsen approach cost matrix analysis

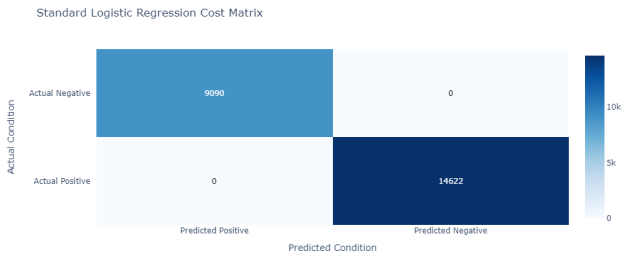

Figure 5.6: Standard logistic regression cost matrix analysis

5.4 | Savings Calculation Explanation

The `calculate_savings` function compares the cost efficiency between standard logistic regression and another classification technique.

5.4.1 | Function Details

- **Purpose**: To quantify the economic benefit of using an alternative technique over standard logistic regression in terms of cost savings.
- **Parameters**:
  - `cost_lr`: The total misclassification cost from using standard logistic regression.
  - `cost_x`: The total misclassification cost from using the alternative technique.

5.4.2 | Savings Calculation

The savings are calculated using the formula:

$$\text{Savings} = \frac{\text{CostLR} - \text{CostX}}{\text{Cost}_{\text{LR}}}$$

- **Interpretation**: The savings score represents the proportion of costs saved by using the new technique. A higher score indicates greater savings.

Table 5.3: Comparison of F1 Scores for Logistic Regression Models

Model Description	F1 Score
Standard Logistic Regression	0.7636
Bahnsen Approach Cost-Sensitive LR	0.4600
Nikou Gunnemann's Approach Variant A	0.7564
Nikou Gunnemann's Approach Variant B	0.1850
Nikou Gunnemann's Approach Variant C	0.7664
Nikou Gunnemann's Approach Variant D	0.7668

Figure 5.7: Cost saving compared to standard logistic regression

6 | References

[1] Günnemann, Nikou Pfeffer, Juergen. (2017). Cost Matters: A New Example-Dependent Cost-Sensitive Logistic Regression Model. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 210-222. 10.1007/978-3-319-57454-7$_1$7.

[2] https://drive.google.com/file/d/1m4P709nIg97gFwG_GnBW8pDH7oEUBxNl/view