आई आई टी हैदराबाद
IIT Hyderabad

CS6890
Fraud Analytics

Example-dependent Cost-sensitive regression

V Harikrishnan
CS23MTECH11008

Suryansh Gautam
CS23MTECH11020

Patel Heetkumar D.
CS23MTECH11029

Anil kumar Sharma
CS23MTECH13001

KR Anuraj
CS23MTECH13002

1. Problem Statement

One of the challenging problem in machine learning area is class imbalance which result model to be biased and predicts wrong result or we can say high false negative rates.

One of the method to solve this problem face due to class imbalance is Cost-Sensitive Learning, where it takes misclassification costs into the consideration which result in better prediction by dealing different misclassification cost separately.

Example-dependent cost-sensitive classification considers varying misclassification costs across examples. It acknowledges that the cost of approving a “bad” samples is different from the cost of declining a “good” samples. These costs can differ due to factors like credit line amounts, loan terms, and interest rates.

2. Description of the Dataset

Given dataset consist of 10 column (A-K) of independent variable, and 1 column (L) of dependent variable.

False Negative rate is given in column (M) which varies from row-to-row as per the risk parameters.

True Positive rate is 6.

False Positive rate is 6.

True Negative rate is 0.

3. Algorithm Used

A. Bahnsen Approach

i. Bahnsen Accuracy

The **Input** of the algorithm are :

1. weights w : It is randomly generated from a range of -10^5 to 10^5.

The **Output** of the algorithm are :

1. Loss : Loss evaluated using formula given by Bahnsen.

Algorithm Bahnsen Accuracy

```
Input : weights w
Output : loss l
1. Initialize the basic logistic regression model.
2. Initialize the model weights with the corresponding inputs weights.
3. Pass the samples of dataset through the logistic model that is created above and
   store the result in variable y_pred.
4. calculate the loss as follows:
   $\text{a} = (\text{labels} \cdot (\text{y\_pred} \times 6 + (1 - \text{y\_pred}) \times \text{fncs})).sum()$
   $\text{b} = ((1 - \text{labels}) \cdot (\text{y\_pred} \times 6)).sum()$
   $\text{loss} = \frac{\text{a+b}}{\text{y\_pred.size}(0)}$
5. return l
```

ii. Genetic algo for Bahnsen

The **Input** of the algorithm are :

1. Bahnsen Accuracy Function : Create a logistic regression model and calculate the Bahnsen loss.

The **Output** of the algorithm are :

1. model : Trained logistic model with optimized Bahnsen loss

---

Genetic algo for Bahnsen

Input : Bahnsen Accuracy Function

Output : model

1. Create default population size amount of inputted Bahnsen models.
2. Train over each of the Bahnsen models.
3. Have a cross-over of the model weights and create 20 child models.
4. Repeat steps 2 - 3 with the newly created child models.
5. return model with optimized Bahnsen loss.

B. Nikou Gunnemann Approach

i. Nikou Gunnemann Accuracy

The **Input** of the algorithm are :

1. weights w : It is randomly generated from a range of -10^5 to 10^5.

The **Output** of the algorithm are :

1. Loss : Loss evaluated using formula given by Nikou Gunnemann.

Algorithm Nikou Gunnemann Accuracy

Input : weights w

Output : loss l

1. Initialize the basic logistic regression model.
2. Initialize the model weights with the corresponding inputs weights.
3. Pass the samples of dataset through the logistic model that is created above and store the result in variable y_pred.
4. calculate the loss as follows:

$$a = (\text{labels} \cdot (-\log(\text{y\_pred} + \epsilon) \times 6 + (-\log(1 - \text{y\_pred} + \epsilon)) \times \text{fncs})).sum()$$

$$b = ((1 - \text{labels}) \cdot (-\log(\text{y\_pred} + \epsilon) \times 6)).sum()$$

$$\text{loss} = \frac{\text{a+b}}{\text{y\_pred.size}(0)}$$

5. return l

ii. Genetic algo for Nikou Gunnemann

The **Input** of the algorithm are :

1. Nikou Gunnemann Accuracy Function : Create a logistic regression model and calculate the Nikou Gunnemann loss.

The **Output** of the algorithm are :

1. model : Trained logistic model with optimized Nikou Gunnemann loss

---

Genetic algo for Nikou Gunnemann

Input : Nikou Gunnemann Accuracy Function

Output : model

1. Create default population size amount of inputted Nikou Gunnemann models.
2. Train over each of the Nikou Gunnemann models.
3. Have a cross-over of the model weights and create 20 child models.
4. Repeat steps 2 - 3 with the newly created child models.
5. return model with optimized Nikou Gunnemann loss.

4. Results

A. Bahnsen Approach

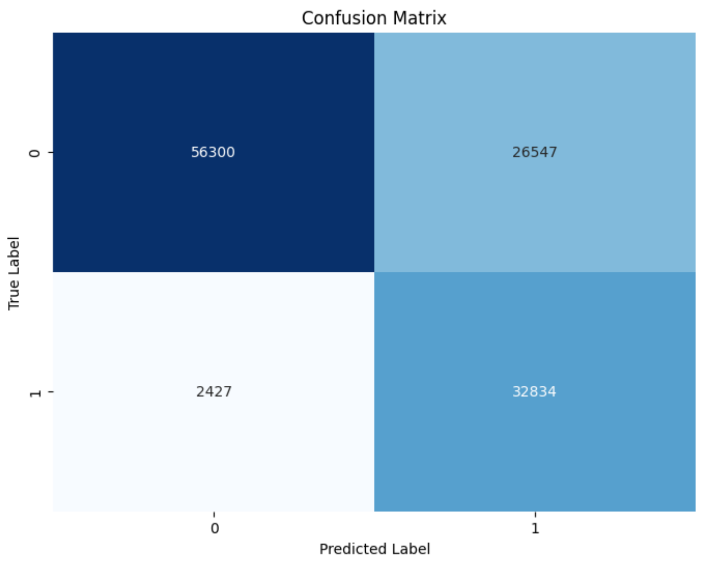

Figure 1: Confusion matrix Bahnsen Approach

B. Nikou Gunnemann Approach

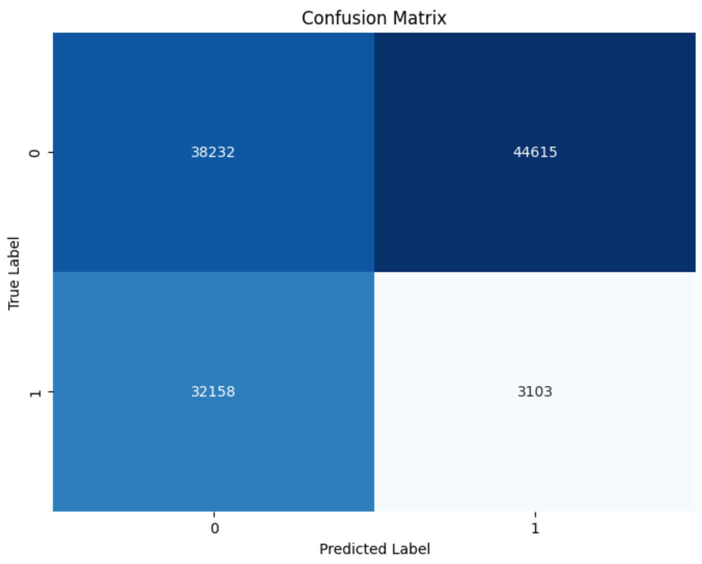

Figure 2: Confusion matrix Nikou Gunnemann Approach