BAREKENG: Journal of Mathematics and Its Applications
December 2026 Volume 20 Issue 4 Page 3533-3546
P-ISSN: 1978-7227 E-ISSN: 2615-3017
https://doi.org/10.30598/barekengvol20iss4pp3533-3546

SCALABLE DETECTION OF COST INEFFICIENCY IN BPJS KESEHATAN CLAIMS USING XGBOOST WITH IMBALANCE LEARNING AND VAEX-BASED BIG DATA PROCESSING

Alpian Roymundus Siringo-ringo[1*], **Ramadhan Paninggalih**[2], **Rizal Kusuma Putra**[3]

[1,2,3]*Departement of Informatics, Faculty of Electrical Engineering, Informatics, and Business, Institut Teknologi Kalimantan*
Jln. Soekarno Hatta No.KM 15, Balikpapan, 76127, Indonesia

*Corresponding author's e-mail: * **arsiringoringo@gmail.com***

Article Info

Article History:

Received: 19th November 2025
Revised: 21st April 2026
Accepted: 05th June 2026
Available online: 24th August 2026

Keywords:

BPJS kesehatan;
Classification;
XGBoost;
Vaex.

ABSTRACT

BPJS Kesehatan Indonesia's national health insurance system, faces significant challenges in managing large-scale claim data, particularly in identifying cost inefficiencies such as abnormal claims, duplication, and misuse. Although machine learning methods have been widely applied in healthcare analytics, limited studies address both large-scale data processing and class imbalance issues in inefficiency detection. This study aims to develop a scalable classification model using XGBoost integrated with Vaex for efficient big data processing. The dataset was obtained from the JKN Healthkathon, consisting of millions of claim records with predefined inefficiency labels. Data preprocessing was performed using Vaex to enable out-of-core computation, followed by model development using XGBoost with various data splitting strategies and imbalance handling techniques, including random oversampling and undersampling. Model performance was evaluated using accuracy, precision, recall, and F1-score. The results indicate that the balanced split with random oversampling achieved the most stable performance, with precision, recall, and F1-score of 0.91. In contrast, stratified splitting without imbalance handling yielded higher accuracy but lower recall. This study contributes by proposing a scalable and efficient framework that integrates XGBoost with Vaex for large-scale healthcare claim analysis and systematically evaluates the impact of imbalance handling strategies on model performance. However, the use of predefined labels with undisclosed generation processes may introduce potential bias in the results.

How to cite this article:

A. R. Siringo-ringo, R. Paninggalih and R. K. Putra., "SCALABLE DETECTION OF COST INEFFICIENCY IN BPJS KESEHATAN CLAIMS USING XGBOOST WITH IMBALANCE LEARNING AND VAEX-BASED BIG DATA PROCESSING", *BAREKENG: J. Math. & App.*, vol. 20, no. 4, pp. 3533-3546, Dec, 2026.

Journal homepage: *https://ojs3.unpatti.ac.id/index.php/barekeng/*
Journal e-mail: *barekeng.math@yahoo.com; barekeng.journal@mail.unpatti.ac.id*

Research Article · **Open Access**

1. INTRODUCTION

Health care services are a fundamental right that every citizen must receive, and the Indonesian government has demonstrated its commitment through the establishment of the Social Security Administering Body for Health (BPJS Kesehatan), which manages the National Health Insurance (JKN) program. As a form of commitment to providing accessible and equitable health services, the program has grown since its implementation in 2014 to become one of the largest health insurance schemes in the world. As November 30, 2024, it covers more than 277 million people, equivalent to 96.8% of Indonesia's population. The program provides essential support to low- and middle-income groups by reducing the cost of care and hospitalization. It adopts a mutual cooperation (*gotong royong*) principle, where all citizens collectively contribute to a mass health insurance system [1].

Despite its extensive benefits, BPJS Kesehatan faces serious challenges, particularly in budget management and long-term financial sustainability. One of the major issues is the persistent deficit in the social security fund. BPJS Kesehatan experience significant deficits between 2014 and 2018, with the gap between revenue and expenditure in the BPJS Kesehatan reaching IDR 11,69 trillion in 2018 [2]. A key factor contributing to this deficit is cost inefficiency in claim processing, which may arise from administrative errors, duplicate claims, overutilization of health services, or fraudulent practices. To address these vulnerabilities, the Indonesian government issued Ministry of Health Regulation No. 16 of 2019 concerning the prevention and handling of fraud in the JKN program, identifying high-risk areas such as false claims and collusion between BPJS personnel and healthcare providers.

Based on the BPJS Kesehatan Program and Financial Management Report, JKN membership continues to increase annually. In 2020, participation reached 222.461.906 individuals, rising to 235.719.262 in 2021 and 267.311.566 in 2023 [3]. As of November 30, 2024, the number of JKN participants reached 277.859.856 individuals. This rapid growth is accompanied by a substantial increase in the volume of submitted claims, creating major challenges for claim verification. The current manual verification system is not effective for handling data at such scale; it is time-consuming and prone to failing to detect patterns of cost inefficiency. Therefore, machine learning approaches capable of processing large claim datasets quickly and accurately are urgently needed to help BPJS Kesehatan optimize budget allocation and prevent inefficiencies.

In this study, cost inefficiency is operationally defined as claims that deviate from standard medical necessity or administrative protocols, which may not necessarily involve the criminal intent characteristic of fraud. Identifying these indications, represented as Label 1 in the dataset is critical because the financial impact of a False Negative (failing to detect an actual inefficiency) is significantly higher than a False Positive (flagging a valid claim for review). Thus, achieving high recall is a primary objective in this application. One machine learning method suitable for such large-scale classification is XGBoost. Nugraha and Irawan [4] employed XGBoost to detect fraud in health insurance claims and achieved a balanced accuracy of 0.98 and a recall of 0.984. Jerith et al. [5] applied XGBoost for predicting the onset of diabetes, achieving an evaluation f1-score of 86.5%. These studies demonstrate the strong potential of XGBoost as a reliable classification model. However, implementing XGBoost on extremely large datasets poses computational challenges, as claim data may contain millions of rows within a single period. To overcome this issue, this study adopts Vaex, a Python library specifically designed for large-scale data processing. Vaex uses out-of-core computation and memory mapping, enabling data to be processed in chunks without fully loading it into RAM. This significantly reduces memory consumption and improves execution time, making it highly suitable for big-data analysis. According to Mozzillo et al. [6], Vaex is particularly effective for workflows involving numerous column-wise operations.

Despite the promising performance of XGBoost in prior studies, several gaps remain. Existing research has not sufficiently explored the integration of scalable data processing frameworks with machine learning models in the context of large-scale healthcare claim analysis. Moreover, the impact of data splitting strategies and class imbalance handling on model performance has not been systematically evaluated. Therefore, this study investigates how XGBoost can be effectively applied to detect cost inefficiency in large-scale BPJS Kesehatan claim data while addressing class imbalance and ensuring computational efficiency.

This study aims to develop and evaluate a classification model for detecting cost inefficiency in BPJS Kesehatan claims by integrating XGBoost with Vaex-based data processing. The contributions of this study are threefold: proposing a scalable framework that combines XGBoost and Vaex for handling large-scale healthcare claim data, systematically evaluating the impact of different data splitting strategies and class imbalance handling techniques, and providing empirical insights into cost inefficiency detection using real-

world BPJS Kesehatan datasets. Compared to prior work, this study not only emphasizes predictive performance but also highlights scalability and robustness in handling imbalanced large-scale datasets.

2. RESEARCH METHODS

2.1 Research Framework

This study adopts a structured research workflow, as shown in Fig. 1, to ensure a systematic and reproducible modeling process. The workflow consists of data collection, preprocessing, feature engineering, data splitting, and class imbalance handling, followed by model development using XGBoost and performance evaluation. The design of this workflow is inspired by general data mining processes and adapted to handle large-scale healthcare claim data efficiently.

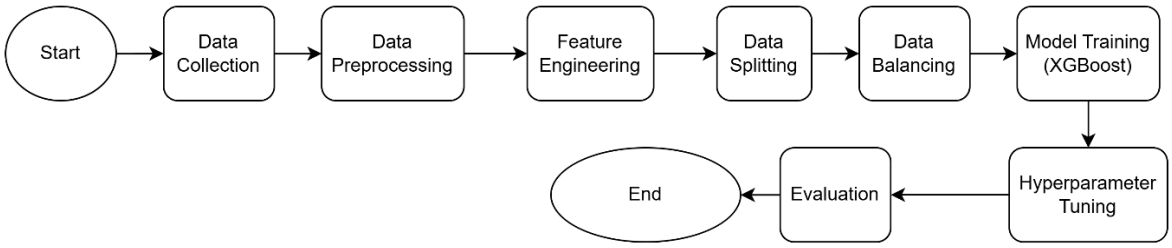

Figure 1. Research Framework

2.2 Machine Learning

Machine learning is a subset of Artificial Intelligence that enables systems to learn patterns from data and make decisions or predictions without explicit programming [7]. It is widely used in data-driven applications due to its strong connections with statistics and data analysis. Based on the learning approach, machine learning is categorized into supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning [8]. This study focuses on supervised learning, particularly classification, where the model learns from labeled data to predict the category of unseen instances.

2.3 eXtreme Gradient Boosting (XGBoost)

A tree is a non-linear data structure primarily used to represent hierarchical relationships between its elements [9]. This tree concept is adapted into a decision-making method through a model known as the Decision Tree. Decisions trees are of the most fundamental models in machine learning for classification and regressions tasks. A decision tree creates a model in the form of a tree structure, where each internal node splits data based on a feature condition, leading to a prediction at the leaf nodes. Despite being intuitive and easy to interpret decision trees are prone to overfitting and low generalization when used independently [10]. To overcome these limitations, ensemble learning is used. It combines multiple base learners to produce a more robust model. In this context, boosting is an effective ensemble method where weak learners-typically shallow decision trees-are trained sequentially [11]. Each new tree aims to reduce the errors of the previous one by giving more weight to misclassified samples [12]. Unlike bagging, which reduces variance, boosting focuses on minimizing bias.

XGBoost (eXtreme Gradient Boosting) is a high-performance machine learning algorithm based on the Gradient Boosting Decision Tree (GBDT) framework [13]. It was introduced by Chen and Guestrin [14] as a scalable and efficient solution for both regression and classification problems. XGBoost enhance traditional gradient boosting by introducing system-level and algorithmic improvements, such as regularization, sparse-aware learning, and parallelization [15].

In classification task, XGBoost works by constructing an ensemble of decision trees in an additive and sequential manner. Each tree is trained to minimize the errors made by the ensemble of previously added trees. The model iteratively optimizes an objective function composed of a training loss and a regularization term, as expressed in the following formula:

$$Obj(\theta) = L(\theta) + \Omega(\theta) \tag{1}$$

Where $L(\theta)$ is the training loss measuring prediction error, and $\Omega(\theta)$ is the regularization term that controls model complexity [16]. The training loss is calculated as:

$$L(\theta) = \sum_{i=1}^{n} l(y_{i,}\hat{y}_{i,}), \tag{2}$$

where y_i is the true label, and $\hat{y}_{i,}$ is the predicted value. For classification, this is often instantiated using cross-entropy loss:

$$L(\theta) = -[y_{i,} \cdot \log(\hat{y}_i) + (1 - y_{i,}) \cdot \log(1 - \hat{y})]. \tag{3}$$

The regularization term is given by:

$$\Omega(\theta) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j{}^2. \tag{4}$$

Each prediction is updated additively:

$$\hat{y}_i{}^{(t)} = \hat{y}_i{}^{(t-1)} + f_t(x_i), \tag{5}$$

where f_t is a new decision tree fitted at iteration t.

To optimize the objective function, XGBoost employs a second-order Taylor expansion, leading to the simplified formulation:

$$Obj(t) = \sum_{i=1}^{n} [g_i f_t(x_i) + \frac{1}{2} h_i f_t{}^2(x_i)] + \Omega(f_t), \tag{6}$$

where g_i and h_i are the first and second derivatives of the loss with respect to $\hat{y}_i$, known as the gradient and Hessian respectively.

The tree structure is defined by assigning each instance x_i to a leaf using a function $q(x)$, and leaf weight are stored in a vector w:

$$f_t(x) = w_{q(x)}, w \in R^t. \tag{7}$$

This enables the regularized objective to be rewritten and optimized in closed form for each split. The final prediction is the sum of predictions from all trees:

$$\hat{y}_i = \sum_{k=1}^{t} f_k(x_i),\ f_k \in F. \tag{8}$$

One of the strengths of XGBoost lies in its ability to compute feature importance using gain, calculated as:

$$gain = \frac{1}{2}\left[\frac{G_L{}^2}{H_L + \lambda} + \frac{G_R{}^2}{H_R + \lambda} + \frac{(G_L + G_R)^2}{H_L + H_R + \lambda}\right] - \gamma, \tag{9}$$

where G_L, G_R, H_L *and* H_R are the accumulated gradients and Hessians for the left and right child nodes.

XGBoost supports various hyperparameters to control model complexity and learning behavior, including *learning_rate, max_depth, min_child_weight, subsample, colsample_bytree, reg_alpha,* and, *gamma.* Proper tuning of these paramaters significantly affects classifications performance [13].

2.4 Vaex

Vaex is a Python library specifically designed for high-performance processing of large tabular datasets that do not fit entirely into memory [17]. Unlike traditional libraries such as Pandas, which load data into RAM, Vaex uses an out-of-core processing model with memory mapping techniques. This allows Vaex to efficiently handle datasets containing millions or even billions of rows without exhausting system resources [18]. One of Vaex's key advantages is its ability to perform fast, lazy computations on large dataframes. It executes operations only when needed, minimizing memory consumption and execution time.

Acccording to [6], Vaex significanly reduces preprocessing time and is well-suited for data workflows requiring efficient colums-wise operations, such as in classification tasks with high dimensional healthcare data. By integrating Vaex with the XGBoost model, this research archives scalable data preprocessing and efficient training on large-scale national health insurance data.

2.5 Grid Search

Grid search is one of the most widely used methods for hyperparameter optimization in machine learning. It systematically explores predefined combinations of hyperparameters to identify the optimal configuration. Although simple, easy to implement, and compatible with parallel processing, its computational cost increases significantly as the number of evaluated hyperparameters grows [19]. This method is considered exhaustive because it evaluates each parameter by testing predefined classification values and reporting performance scores for every configuration. Grid search achieves its maximum effectiveness when the upper and lower bounds of each parameter are well defined [20].

2.6 Data Collection

The dataset used in this research was sourced from the 2022 Healthkathon organized by BPJS Kesehatan, which is publicly accessible via the JKN Data Portal and managed by the Public Information and Documentation Management (PPID) of BPJS Kesehatan. The dataset simulates national health insurance claims from referral health facilities and contains millions of rows representing patient visits and associated medical records. In addition, the dataset includes a predefined binary label indicating potential cost inefficiency (0 = non-inefficient, 1 = inefficient), which is directly provided by the dataset source. This label is used as the target variable in the supervised learning process, reflecting real-world classification scenarios based on prior domain-specific identification.

2.7 Data Preprocessing

Data preprocessing is a key step in preparing data for machine learning models. It involves data cleaning, feature engineering, and addressing issues found during exploratory analysis. The goal is to ensure the dataset is clean, structured, and ready for XGBoost classification. Initial steps included identifying missing values and inconsistencies, particularly in attributes like patient gender, medical service type, and discharge condition. Missing values were imputed or removed based on their impact on model quality. Feature engineering included encoding categorical variables and converting temporal data into numeric duration features.

2.8 Data Splitting

Data splitting is the process of dividing a dataset, typically into training data and testing data. This process is crucial in the development of a machine learning model. The training data is used to train the model, while the testing data is used to evaluate the model's performance after the training phase is completed. The ratio of the split is usually adjusted based on the size of the dataset. There are several approaches that can be used for data splitting, such as balanced split and stratified split [21].

2.9 Data Balancing for Data Train

Data balancing is the process of equalizing the distribution of class labels in a dataset, particularly when class imbalance occurs. Class imbalance can lead to model bias toward the majority class. Several techniques can be used to balance the data, including oversampling (increasing samples from the minority class) and undersampling (reducing samples from the majority class). This process is essential to ensure the model can accurately predict the minority class. In this study, each data balancing technique was combined with different data splitting approaches. The techniques used were random oversampling and random undersampling.

2.10 Model Training and Hyperparameter Tuning

The XGBoost algorithm was used to train the classification model due to its efficiency and high performance on large datasets. Two key hyperparameters were tuned to improve model accuracy: *learning_rate* and *max_depth.* The tested values are shown in Table 1 below:

Table 1. Hyperparameter Values

Parameter	Value	
learning_rate	0.05	0.1
max_depth	3	5

The *learning_rate* controls how quickly the model learns, while *max_depth* limits tree complexity to prevent overfitting. These parameters were tested across different data splitting and class balancing scenarios to identity the best-performing model configuration.

2. 11 Evaluation Metrics

To assess the classification performance of the XGBoost model, this research uses several evaluation metrics derived from confusion matrix, which compres the predicted and actual class labels [22].

3. RESULTS AND DISCUSSION

3.1 Data Preprocessing

The data preprocessing stage in this study was carried out to ensure that the dataset was clean, structured, and ready for modelling. The dataset used consisted of three related files: *sampling_healthkathon,* which contains data on patient visits to referral health facilities; *sampling_healthkathon_diagnosa,* which contains disease diagnosis codes according to ICD-10; and *sampling_healthkathon_procedure,* which contains medical procedure codes based on ICD-9 CM. these datasets were merged using the *id* column as a foreign key. Table 2 presents a detailed explanation of the dataset related to JKN participant visits.

Table 2. Description of the *Sampling_Healthkathon* Dataset

No	Attribute Name	Rows with Data	Rows with Missing Values
1.	*id*	11401882	0
2.	*id_peserta*	11401882	0
3.	*dati2*	11401882	0
4.	*typefaskes*	11401882	0
5.	*usia*	11401882	0
6.	*jenkel*	11401833	49
7.	*pisat*	11401692	190
8.	*tgldatang*	11401882	0
9.	*tglpulang*	11401882	0
10.	*jenispel*	11401882	0
11.	*politujuan*	7360427	4041455
12.	*diagfktp*	11399352	2530
13.	*biaya*	11344067	57815
14.	*jenispulang*	11401843	39
15.	*cbg*	11401882	0
16.	*kelasrawat*	11401882	0
17.	*kdsa*	14883	11386999
18.	*kdsp*	74517	11327365
19.	*kdsr*	8538	11393344
20.	*kdsi*	11359	11390523
21.	*kdsd*	154090	11247792
22.	*label*	11401882	0

During exploratory data analysis, several columns were identified to contain significant missing values, especially in the procedure and diagnosis datasets. Columns such as *kdsa, kdsp, kdsr, kdsi* and, *kdsd* were were dropped because more than 90% of their values were missing. This threshold was chosen because features with such extreme missingness do not provide sufficient information for the model to learn and could

potentially introduce noise. Columns like *biaya, jenkel,* and *politujuan,* were imputed using statistical techniques such as mean or mode. A new feature was also engineered: *durasi_rawat,* which calculates the length of hospital stay based on *tgldatang* and *tglpulang.* Fig. 2 illustrates the number of labels in the dataset, showing a significant class imbalance. Table 3 Describes the changes made to the *diagfktp* and *diag columns*. Table 4 shows the changes applied to the proc column.

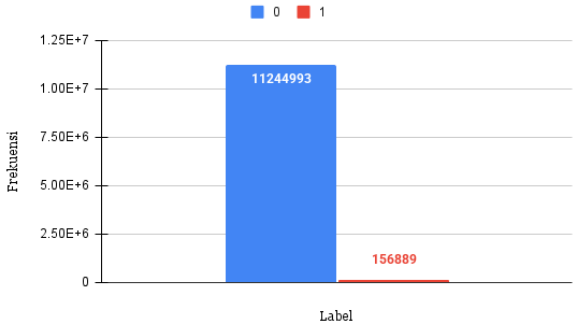

Figure 2. Number of Labels in The Sampling_Healthkathon Dataset

Table 3. Disease Diagnosis Grouping

No	Code Range	Diagnosis Grouping
1.	00	Certain infectious and parasitic diseases
2.	C00 – D48	Neoplasms
3.	D50 - D89	Blood and blood-forming organs diseases and disorders involving immune mechanism
4.	E00 – E90	Endocrine, nutritional and metabolic diseases
5.	F00 – F99	Mental and behavioural disorders
6.	G00 – G99	Diseases of the nervous system
7.	H00 – H59	Diseases of the eye and adnexa
8.	H60 – H95	Diseases of the ear and mastoid process
9.	I00 – I99	Diseases of the circulatory system
10.	J00 – J99	Diseases of the respiratory system
11.	K00 – K93	Diseases of the digestive system
12.	L00 – L99	Diseases of the skin and subcutaneous tissue
13.	M00 – M99	Diseases of the musculoskeletal system and connective tissue
14.	N00 – N99	Diseases of the genitourinary system
15.	O00 – O99	Pregnancy, childbirth and the puerperium
16.	P00 – P96	Certain conditions originating in the perinatal period
17.	Q00 – Q99	Congenital malformations, deformations and chromosomal abnormalities
18.	R00 – R99	Symptoms, signs and abnormal clinical and laboratory findings, not elsewhere classified
19.	S00 – T98	Injury, poisoning and certain other consequences of external causes
20.	V09 – Y98	External causes of morbidity and mortality
21.	Z00 – Z99	Factors influencing health status and contact with health services
22.	U00 – U85	Codes for special purposes
23.	Value out of range	Uncategorized
24.	Invalid code	Invalid

Table 4. Disease Procedure Grouping

No	Code Range	Procedure Grouping
1.	00	Procedures and Interventions, Not Elsewhere Classified
2.	01 – 05	Operations on the Nervous System
3.	06 – 07	Operations on the Endocrine System
4.	08 – 16	Operations on the Eye
5.	17	Other Miscellaneous Diagnostic and Therapeutic Procedures
6.	18 – 20	Operations on the Ear
7.	21 – 29	Operations on the Nose, Mouth, and Pharynx
8.	30 – 34	Operations on the Respiratory System
9.	35 – 39	Operations on the Cardiovascular System
10.	40 – 41	Operations on the Hemic and Lymphatic System
11.	42 – 54	Operations on the Digestive System
12.	55 – 59	Operations on the Urinary System
13.	60 – 64	Operations on the Male Genital Organs
14.	65 – 71	Operations on the Female Genital Organs
15.	72 – 75	Obstetrical Procedures

No	Code Range	Procedure Grouping
16.	76 – 84	Operations on the Musculoskeletal System
17.	85 – 86	Operations on the Integumentary System
18.	87 – 99	Miscellaneous Diagnostic and Therapeutic Procedures

Feature transformation was performed to handle categorical variables while strictly preventing data leakage. To ensure an unbiased evaluation, all encoders were fitted exclusively on the training set, and the learned mappings were subsequently applied to the test set. Columns such as *typefaskes*, *jenkel*, *cbg1*, *diagfktp*, *diag*, and *proc* were transformed using LabelEncoder. For the *politujuan* column, Frequency Encoding was implemented by calculating category frequencies based solely on the training distribution to prevent information from the test set from influencing the feature space. This approach ensures that the model's performance metrics reflect its ability to generalize to unseen data.

1. Label encoding was applied to: *typefaskes, jenkel, cbg1, diagfktp, diag, proc.*
2. Frequency encoding was applied to: *politujuan.*

Table 5 presents the transformation of categorical data into numerical data.

Table 5. Categorical Columns after Data Transformation

Typefaskes	*Jenkel*	*Cbg1*	*Diagfktp*	*Diag*	*proc*	*politujuan*
9	1	4	12	6	1	0.06491882593757517
9	1	4	13	5	1	0.14490386071408087
…	…	…	…	…	…	…
12	1	6	15	15	15	0.02793260821936121
6	1	4	9	9	1	0.08569271820364505

Finally, the three datasets were integrated, ensuring in categorical values. Continuous variables such as *biaya* were normalized using logarithmic transformation to reduce skewness and improve model convergence. The final processed datasets was compiled into a single file, ready for model training. Table 6 presents the final dataset that will be used for model development

Table 6. Dataset after the Data Preparation

No	Columns Name	Unique Value	Description
1.	*dati2*	488	Healthcare facility location
2.	*typefaskes*	26	Type of healthcare facility
3.	*usia*	111	Participant's age
4.	*jenkel*	2	Participant's gender
5.	*pisat*	5	Participant relationship status
6.	*durasi_rawat*	128	Participant's treatment duration
7.	*jenispel*	2	Type of service
8.	*politujuan*	235	Target clinic
9.	*diagfktp*	24	Diagnosis from primary care (FKTP)
10.	*biaya*	13787	Cost
11.	*jenispulang*	5	Participant's condition at discharge
12.	*cbg1*	23	CMG (Casemix Main Groups)
13.	*cbg2*	9	Type of case group
14.	*cbg3*	61	Case group specification
15.	*kelasrawat*	3	Care class
16.	*diag*	23	Participant's diagnosis
17.	*levelid*	2	Type of diagnosis
18.	*proc*	19	Procedure code
19.	*label*	2	Potential inefficiency label

3.1.1 Comparison of Processing Time Using Pandas and Vaex

Data processing and model training were performed using two libraries: Vaex and Pandas, compared based on their execution time. Each experiment was executed only once under the same computer conditions, with no other tasks running. All computational tasks, including data preprocessing using Vaex and model training with XGBoost, were conducted on a machine equipped with an AMD Ryzen 5 5000U processor (2.1 GHz, 6 cores), integrated Radeon Graphics, and 16 GB of RAM. Fig. 3 shows a comparison of Vaex and Pandas.

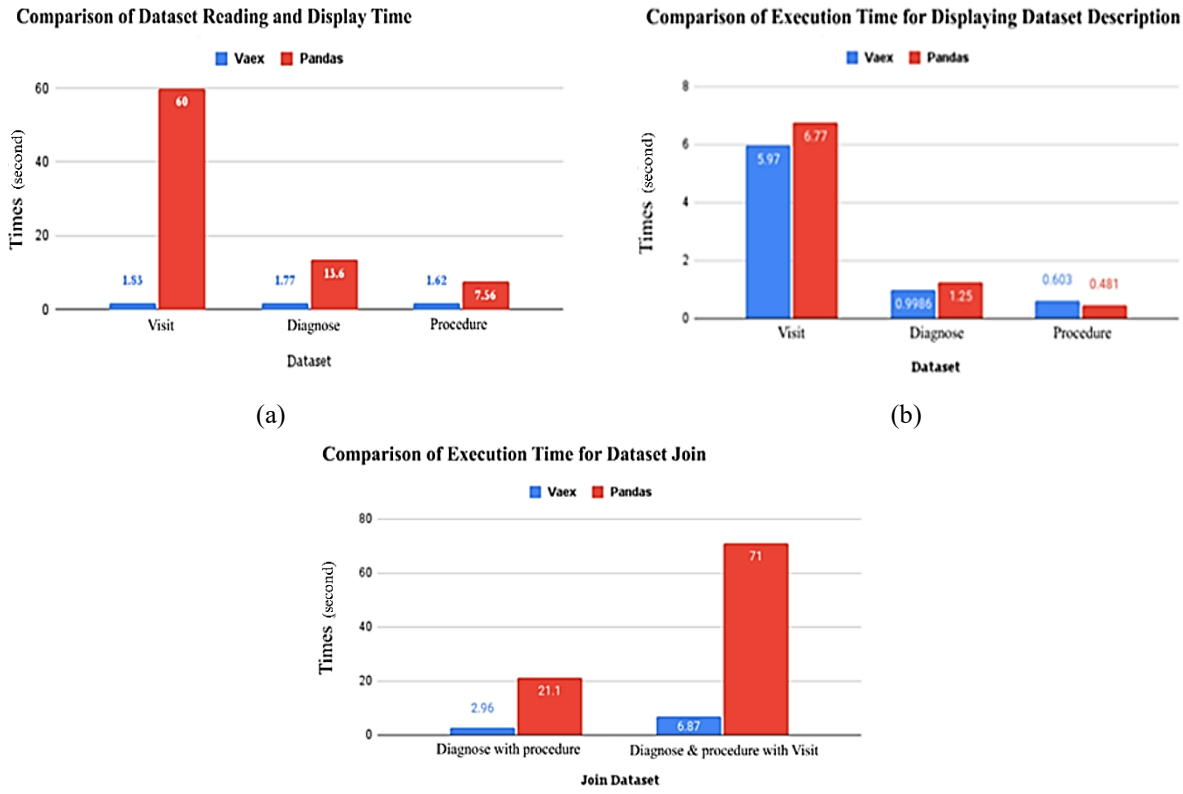

Figure 3. Comparison of Vaex and Pandas by Display Time
(a) Comparison of Dataset Reading and Display Time,
(b) Comparison of Execution Time for Displaying Dataset Description,
(c) Comparison of Execution Time for Dataset Join

3.2 Data Splitting

The dataset used in this study contains 9,677,137 records, consisting of two class labels: label "0" with 9,434,245 samples and label "1" with 242,892 samples. The dataset was partitioned using two different strategies: stratified split and balanced split. In the stratified split, an 80:20 ratio was applied while maintaining the original class distribution to evaluate the model's performance under real-world imbalanced conditions. In contrast, the balanced split was specifically designed to create a balanced evaluation environment. This was achieved by taking 20% of the minority class (Label '1') for the test set and matching it with an equal number of majority class samples (Label '0'), resulting in a 1:1 ratio in the testing subset. The remaining data were assigned to the training set. This dual-splitting approach allows for a comprehensive assessment of the model's robustness, comparing its ability to handle original data distributions versus its performance on a controlled, balanced test environment. Table 7 presents the results of data splitting using balanced split and stratified split.

Table 7. Composition of Data Split

Label	Balanced Split		Stratified Split	
	Data Training	Data Testing	Data Training	Data Testing
0	9385667	48578	7547396	1886849
1	194314	48578	194314	48578

Both splitting methods were designed to evaluate how different data distributions impact the model's ability to generalize, especially under class imbalance scenarios.

3.3 Data Balancing

To address the class imbalance issue in the training data, two techniques were applied: random oversampling and random undersampling. These were tested on both the balanced split and stratified split training sets to observe their effects on class distribution. In the random oversampling approach, the number

of samples in the minority class (label 1) was increased to match the majority class (label 0), without removing any data.

3.4 Model Training and Hyperparameter Tuning

To evaluate model performance during training, log loss curves were plotted over *500 boosting rounds* with *early_stopping_rounds*=5. The tested parameter combinations were *learning_rate* = {0.05, 0.1} and *max_depth* = {3, 5}.

3.4.1 Balanced Split

The following graphs present the patterns of change in log loss as the boosting iterations increase, both for the training data and the testing data.

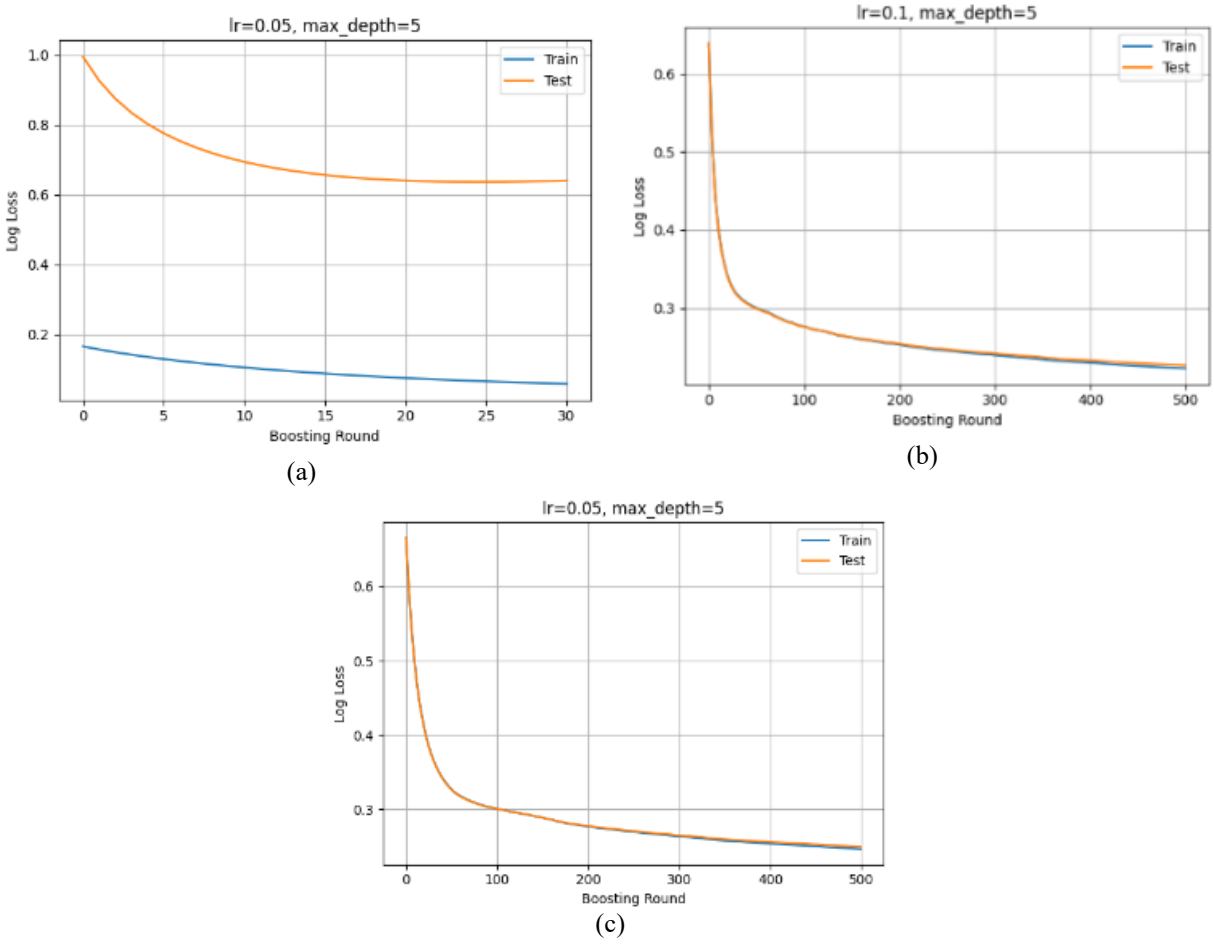

Figure 4. Log Loss Graph of the Model Using Balanced Split
(a) Without Handling Imbalance, (b) Using with Random Oversampling, (c) with Random Undersampling

Fig. 4 illustrate the effect of different data balancing strategies on model performance. Fig. 4 (a) shows that without any balancing strategy, the model closely fit the training data but failed to generalize to unseen data. While training loss decreased, testing performance remained low, indicating that the model mainly learned from the dominant class due to class imbalance. In contrast, Fig. 4 (b) presents the result of applying oversampling, where the model successfully learned both majority and minority classes more equally. This led to the best overall performance, with consistent training and testing curves, and balanced evaluation metrics such as precision and F1-score. Oversampling enhanced minority class representation without removing data from the majority class. Meanwhile, Fig. 4 (c) shows the outcome of random undersampling. Although the model trained stably, performance was slightly lower than the oversampling case due to reduced training data, which limited the model's ability to capture complex patterns.

3.4.2 Stratified Split

The following graphs present the patterns of change in log loss as the boosting iterations increase, both for the training data and the testing data.

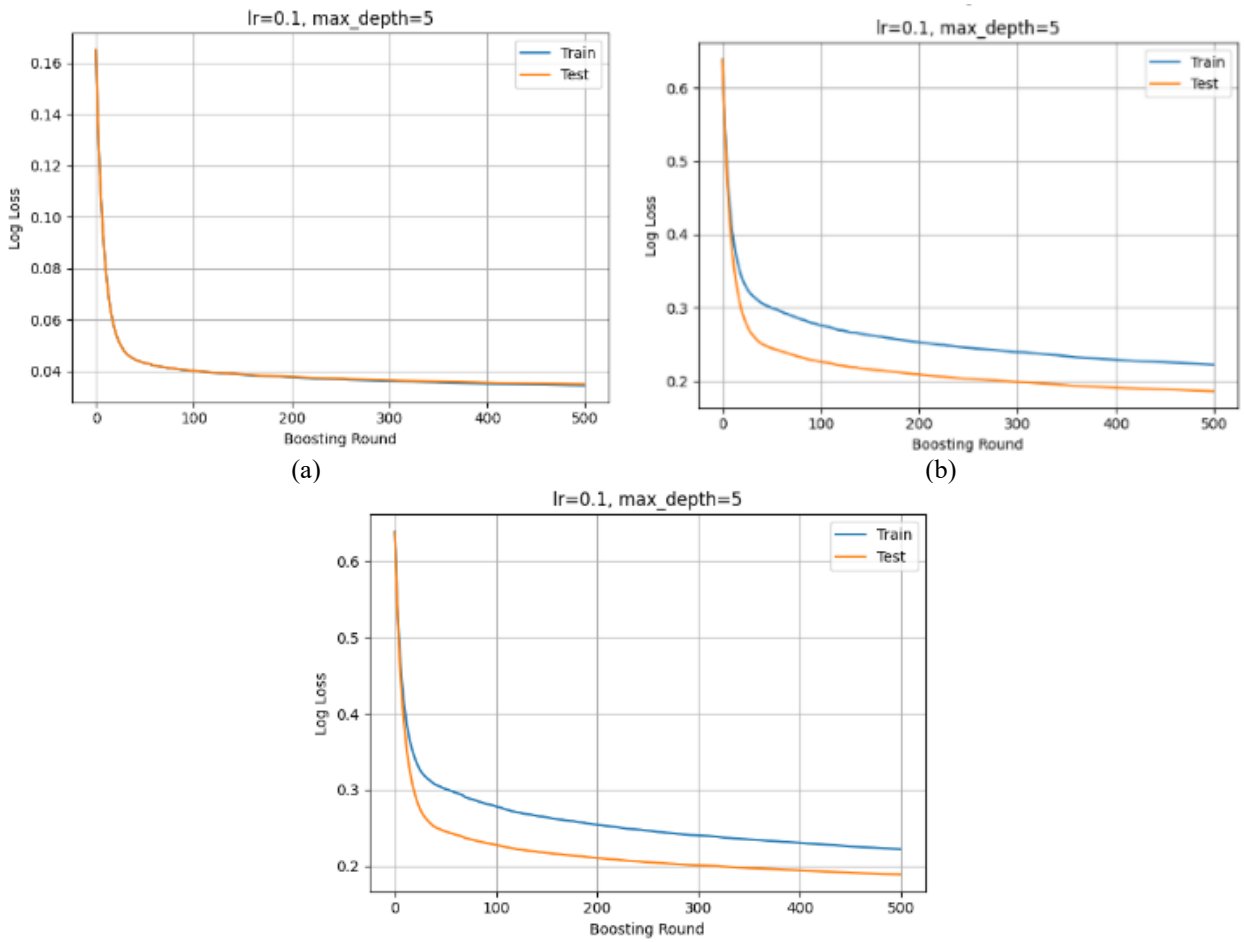

Figure 5. Log Loss Graph of The Model Using Stratified Split
(a) Without Handling Imbalance, (b) Using with Random Oversampling, (c) with Random Undersampling

Fig. 5 highlight the impact of different data balancing approaches on the model's ability to distinguish between classes. Fig. 5 (a) shows that, in the absence of any balancing strategy, the model focused heavily on the majority class, failing to effectively capture the characteristics of the minority class. This led to high performance on the majority class but poor recall and uneven metric distribution overall. Fig. 5 (b) demonstrates how applying oversampling significantly improved the representation of the minority class, resulting in more balanced classification outcomes. This approach led to notable improvements in both recall and F1-score for the minority class compared to the unbalanced stratified scenario. Meanwhile, Fig. 5 (c) presents the effect of random undersampling. Although the model maintained consistent training behavior and prediction stability, the reduced size of the training data limited its ability to learn complex patterns, resulting in relatively lower performance scores. Nonetheless, the model still showed a fair ability to differentiate between the two classes. Overall, the best configuration was found in the balanced split with random oversampling, particularly with *learning_rate* = 0.1 and *max_depth* = 5, as it produced a good balance between accuracy, precision, and sensitivity toward the minority class.

3.5 Evaluation

To provide a fair comparison, this study evaluates the proposed model against previous research conducted on the same dataset. Nugraha and Irawan [3] applied XGBoost for healthcare claim analysis and reported a balanced accuracy of 0.98. Table 8 presents a comparison between previous research and this study. In comparison, the proposed approach achieved an F1-score of 0.91 with improved recall performance under imbalanced conditions. This indicates that the proposed method is more effective in identifying minority class instances, which is critical in detecting cost inefficiency.

Table 8. Best-Performing Experiment Based on Accuracy

Study	Model	Accuracy	Recall	Precision	F1-score
Nugraha and Irawan [3]	XGBoost without Oversampling and Undersampling	0.986	0.35281	0.703	0.672
	XGBoost using Oversampling (SMOTE)	0.951	0.255	0.822	0.124
	XGBoost using Oversampling (ADASYN)	0.953	0.229	0.789	0.117
	XGBoost using Unersampling (Tomek Links)	**0.987**	0.349	0.700	0.665
	XGBoost using Unersampling (NearMiss)	0.539	0.669	0.198	0.384
Proposed Study	XGBoost using Balanced split and random oversampling	0.908	**0.91**	**0.91**	**0.91**

To provide a deeper insight into the model's classification performance and ensure transparency in the evaluation process, Fig. 6 illustrates the confusion matrix for the proposed XGBoost model using balanced split and random oversampling.

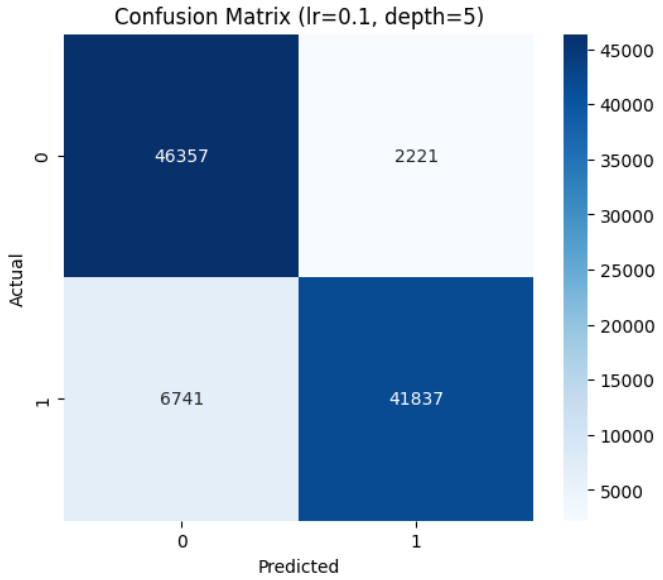

Figure 6. Confusion Matrix for XGBoost using Balanced Split and Random Oversampling

In the context of BPJS Kesehatan claim verification, minimizing False Negatives (FN) is the highest priority to prevent budget leakage. Therefore, the balanced split with random oversampling (learning_rate = 0.1, max_depth = 5) is identified as the most suitable model. Despite a lower global accuracy (0.9078), it achieved a balanced and superior performance with precision, recall, and F1-score all at 0.91. This high recall ensures that the model consistently detects the majority of actual inefficiency cases, making it the most robust choice for practical implementation where financial oversight and classification fairness are critical.

4. CONCLUSION

This study concludes that the integration of the XGBoost model with the Vaex library provides a scalable and efficient framework for detecting cost inefficiency indications in large-scale healthcare claim data. The experimental results demonstrate that the balanced split with random oversampling is the most effective configuration, achieving a balanced precision, recall, and F1-score of 0.91. Although stratified splitting yielded higher global accuracy, it resulted in a lower recall, which is less ideal for identifying rare inefficiency cases in a real-world context. The use of Vaex for data preprocessing was crucial, enabling efficient out-of-core computation for millions of records without being limited by RAM capacity. Despite these promising results, several limitations remain. The research relies on a sample dataset with predefined labels where the specific generation process was undisclosed, potentially introducing bias. Additionally, the study's external validity needs further testing in a live production environment. For future work, it is recommended to implement cost-sensitive learning to further minimize False Negatives and utilize interpretability frameworks like SHAP to provide transparent insights for medical auditors. Validating this model with a wider range of real-time clinical data will also be essential to enhance its robustness and practical utility.

Author Contributions

Alpian Roymundus Siringo-ringo: Conceptualization, Formal Analysis, Investigation, Methodology, Project Administration, Software, Visualization, Writing - Original Draft, Writing - Review and Editing. Ramadhan Paninggalih: Conceptualization, Formal Analysis, Methodology, Resources, Supervision, Validation, Writing - Original Draft, Writing - Review and Editing. Rizal Kusuma Putra: Conceptualization, Formal Analysis, Methodology, Resources, Supervision, Validation, Writing - Original Draft, Writing - Review and Editing. All authors discussed the results collaboratively and contributed to the final version of the manuscript.

Funding Statement

The authors received no financial support for the research, authorship or publication of this article.

Acknowledgment

The authors would like to express their gratitude and appreciation to all those who have already supported the completion of this work.

Declarations

The authors declare no conflicts of interest to report study.

Declaration of Generative AI and AI-assisted technologies

ChatGPT was utilized only to improve the readability and grammatical structure of the manuscript. No AI tool was used to generate or alter the research data, methodology, results, or interpretations. All content was verified by the authors for accuracy and consistency with the study.

REFERENCES

[1] C. F. P. Zebua, D. Ardhila, Y. Yuriska, and F. P. Gurning, "ANALISIS PERAN PROGRAM JAMINAN KESEHATAN NASIONAL (JKN) DALAM MENGURANGI BEBAN FINANSIAL PASIEN: STUDI LITERATURE," *El-Mujtama J. Pengabdi. Masy.*, vol. 4, no. 2, pp. 802–811, August 2023, doi: https://doi.org/10.47467/elmujtama.v4i2.4407.

[2] R. Annisa, S. Winda, E. Dwisaputro, and K. N. Isnaini, "MENGATASI DEFISIT DANA JAMINAN SOSIAL KESEHATAN MELALUI PERBAIKAN TATA KELOLA," *INTEGRITAS J. Antikorupsi*, vol. 6, no. 2, pp. 209–224, December 2020.

[3] BPJS Kesehatan, "PORTAL DATA JAMINAN KESEHATAN NASIONAL," Portal Data BPJS Kesehatan, 2024. [Online].

[4] A. C. Nugraha and M. I. Irawan, "KOMPARASI DETEKSI KECURANGAN PADA DATA KLAIM ASURANSI PELAYANAN KESEHATAN MENGGUNAKAN METODE SUPPORT VECTOR MACHINE (SVM) DAN EXTREME GRADIENT BOOSTING (XGBOOST)," *J. Sains dan Seni ITS*, vol. 12, no. 1, May 2023, doi: https://doi.org/10.12962/j23373520.v12i1.107032.

[5] G. G. Jerith et al., "EVALUATION OF CATBOOST FOR DIABETES PREVENTION IN COMPARISON TO XGBOOST: TO AVERT MORTALITY BY DEVELOPING AN AI MODEL CAPABLE OF PREDICTING THE ONSET OF DIABETES," *2024 Int. Conf. Electron. Syst. Intell. Comput.*, pp. 319–324, Nov. 2024, doi: https://doi.org/10.1109/ICESIC61777.2024.10846686.

[6] A. Mozzillo et al., "EVALUATION OF DATAFRAME LIBRARIES FOR DATA PREPARATION ON A SINGLE MACHINE", [Online]. doi : https://doi.org/10.1002/9781394155408.ch2

[7] G. A. Shafila, "IMPLEMENTASI METODE EXTREME GRADIENT BOOSTING (XGBOOST) UNTUK KLASIFIKASI PADA DATA BIOINFORMATIKA (STUDI KASUS : PENYAKIT EBOLA , GSE 122692)," Bachelor [Thesis] Islamic University of Indonesia., 2020 [Online].

[8] I. H. Sarker, "MACHINE LEARNING: ALGORITHMS, REAL-WORLD APPLICATIONS AND RESEARCH DIRECTIONS," *SN Comput. Sci.*, vol. 2, no. 3, pp. 1–21, March 2021, doi: https://doi.org/10.1007/s42979-021-00592-x.

[9] Annisa, "STRUKTUR DATA TREE," *Fikti Umsu*, pp. 1–10, 2023, [Online].

[10] J. Han, M. Kamber, and J. Pei, *DATA MINING: CONCEPTS AND TECHNIQUES*. 2011. doi: https://doi.org/10.1016/C2009-0-61819-5

[11] N. F. Bakri, *ANALISIS KLASIFIKASI FINANCIAL DISTRESS DENGAN MEMBANDINGKAN METODE EXTREME GRADIENT BOOSTING DAN ARTIFICIAL NEURAL NETWORK MENGGUNAKAN RASIO KEUANGAN PADA PERUSAHAAN PERBANKAN TAHUN 2013–2022,* Bachelor [Thesis] Hasanuddin University., 2024.

[12] J. H. Friedman, "GREEDY FUNCTION APPROXIMATION: A GRADIENT BOOSTING MACHINE," *Ann. Stat.*, vol. 29, no. 5, pp. 1189–1232, November 2001, doi: https://doi.org/10.1214/aos/1013203451 .

[13] A. N. Rachmi, "IMPLEMENTASI METODE RANDOM FOREST DAN XGBOOST PADA KLASIFIKASI CUSTOMER CHURN," pp. 1–101,. Bachelor [Thesis] Islamic University of Indonesia, 2020.

[14] T. Chen and C. Guestrin, "XGBOOST: A SCALABLE TREE BOOSTING SYSTEM," *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, vol. 13-17-Augu, pp. 785–794, 2016, doi: https://doi.org/10.1145/2939672.2939785

[15] S. E. Herni Yulianti, Oni Soesanto, and Yuana Sukmawaty, "PENERAPAN METODE EXTREME GRADIENT BOOSTING (XGBOOST) PADA KLASIFIKASI NASABAH KARTU KREDIT," *J. Math. Theory Appl.*, vol. 4, no. 1, pp. 21–26, 2022, doi: https://doi.org/10.31605/jomta.v4i1.1792.

[16] L. Zhang and C. Zhan, *MACHINE LEARNING IN ROCK FACIES CLASSIFICATION: AN APPLICATION OF XGBOOST*. 2017. doi: https://doi.org/10.1190/IGC2017-351.

[17] A. Mozzillo, "MAXIMIZING EFFICIENCY IN EXISTING DATA PREPARATION PIPELINES," *CEUR Workshop Proc.*, vol. 3478, pp. 720–726, 2023.

[18] M. A. Breddels and J. Veljanoski, "VAEX: BIG DATA EXPLORATION IN THE ERA OF GAIA," *Astron. Astrophys.*, vol. 618, pp. 1–13, October 2018, doi: https://doi.org/10.1051/0004-6361/201732493.

[19] R. Schaer, H. Müller, and A. Depeursinge, "OPTIMIZED DISTRIBUTED HYPERPARAMETER SEARCH AND SIMULATION FOR LUNG TEXTURE CLASSIFICATION IN CT USING HADOOP," *J. Imaging*, vol. 2, no. 2, 2016, doi: https://doi.org/10.3390/jimaging2020019.

[20] M. M. RAMADHAN, I. S. SITANGGANG, F. R. NASUTION, and A. GHIFARI, "PARAMETER TUNING IN RANDOM FOREST BASED ON GRID SEARCH METHOD FOR GENDER CLASSIFICATION BASED ON VOICE FREQUENCY," *DEStech Trans. Comput. Sci. Eng.*, no. cece, 2017, doi: https://doi.org/10.12783/dtcse/cece2017/14611.

[21] A. A. Khan, "BALANCED SPLIT: A NEW TRAIN-TEST DATA SPLITTING STRATEGY FOR IMBALANCED DATASETS" 2022, [Online]. [Accessed: 2 March 2025]

[22] Z. Karimi, *Confusion Matrix*. 2021, [Online].[Accessed: 29 January 2025]