Improving Multilabel Classification Model Performance in Imbalanced Datasets with Group-Level Undersampling

Danny Sebastian
Fakultas Teknologi Informasi, Universitas Kristen Satya Wacana, Salatiga, Indonesia | Informatika, Universitas Kristen Duta Wacana, Yogyakarta, Indonesia
danny.sebastian@staff.ukdw.ac.id (corresponding author)

Hindriyanto Dwi Purnomo
Fakultas Teknologi Informasi, Universitas Kristen Satya Wacana, Salatiga, Indonesia
hindriyanto.purnomo@uksw.edu

Irwan Sembiring
Fakultas Teknologi Informasi, Universitas Kristen Satya Wacana, Salatiga, Indonesia
irwan@uksw.edu

Received: 24 February 2025 | Revised: 18 April 2025 | Accepted: 8 May 2025

 DOI: https://doi.org/10.48084/etasr.10680

ABSTRACT

Multilabel classification presents unique challenges, particularly when dealing with imbalanced datasets. This study introduces a Group-Level Undersampling (GLU) technique designed to enhance the performance and efficiency of multilabel classification models. By converting multilabels into combined labels and categorizing them into group levels, the proposed method preserves the diversity of label combinations, which is crucial for accurate classification. Undersampling is particularly challenging in multilabel classification because removing a single instance can affect the combination of classes. This study utilized Google Maps reviews of tourist sites in Yogyakarta, Indonesia, manually labeled into 11 classes. The IndoBERTweet model was fine-tuned using the GLU generated dataset, and its performance was compared with models trained on randomly selected datasets. Experimental results demonstrate that the GLU method can maintain the variation of multilabel datasets by using combined labels and group levels. The experimental results show that the GLU dataset significantly improves model accuracy and F1 scores while reducing computational time and resources. The findings suggest that the proposed undersampling technique offers a robust solution to address imbalanced datasets in multilabel classification, paving the way for more efficient and accurate machine learning models. However, there are still weaknesses in the ratio calculation formula, allowing some datasets to be suboptimal at certain group levels. Future research should apply the GLU technique to various datasets and multilabel tasks to assess its generalizability. Investigating hybrid data-balancing methods could further enhance model performance and efficiency. A sensitivity analysis on ratioBaseLine values and refining the ratio calculation formula will improve the robustness of the GLU method.

Keywords-multilabel classification; undersampling; group level undersampling; dataset balancing

I. INTRODUCTION

Classification is a key technique in machine learning that aims to categorize data into specific classes based on their features [1]. There are two types of classification: unilabel and multilabel. Unilabel classification assumes that each sample can be assigned only one label or class. For example, in an image classification problem, one image can only be assigned one label, such as "dog" or "cat." On the other hand, multilabel classification allows one sample to have more than one label [2]. For example, in a text recognition task, a document can have multiple labels such as "politics" and "economy" simultaneously. These two types of classifications affect the training strategy of the model.

Numerous challenges exist in the development of both unilabel and multilabel classification models [3]. An imbalanced dataset refers to a disparity in sample sizes between different classes, causing the model to predominantly predict the majority class while neglecting the minority class [1, 4]. The absence of a feature representation to distinguish existing classes makes the model incapable of generating precise

predictions [2]. The presence of irrelevant, erroneous, or confusing data in the training dataset, known as data noise, adversely affects model performance [3, 5, 6]. The subsequent issue is overfitting, which arises when the model retains the training data excessively and does not effectively generalize to the test data [4, 7, 8]. Underfitting occurs when a model is excessively simplistic, failing to adequately distinguish between existing classes and yielding poor predictive results [5, 7, 8]. Data inaccuracies, such as noise, outliers, or class imbalance, can affect the propensity of a model to overfit or underfit [9]. Therefore, it is important to ensure good data quality and perform proper data preprocessing before training the model [10, 11]. The selection of the dataset is important when creating a classification model [12], as it must accurately represent the characteristics of each category/label, and the number of samples within each category/label should be balanced to mitigate the risk of class imbalance.

Several methods have been developed to deal with imbalanced datasets, including oversampling, undersampling, synthetic minority, and class weights [13]. Oversampling increases the number of samples from minority classes by doubling existing samples or adding synthetic samples (Synthetic Minority Over-sampling Technique/SMOTE) [14, 15]. Undersampling reduces the number of samples from the majority class to balance it with the number of samples in the minority classes [15, 16]. Oversampling preserves all information within the dataset, in contrast to undersampling, which eliminates samples [10]. However, oversampling poses the risk of overfitting in minority classes and prolongs computational time due to the influx of new data [17]. This contrasts with undersampling, which offers the benefit of decreased computational time due to the reduction in dataset size. The class weight approach employs a penalization technique when an error occurs in the minority class [18]. This method has the weakness of being sensitive to the presence of outliers and noise in the dataset.

In multilabel classification, the challenge is greater because the model must connect the dependencies between labels that may be related to each other [19]. This approach often uses methods such as classifier chains and binary relevance to address the problem of interlabel interactions, which is different from conventional approaches to unilabel classification [20]. Due to the different characteristics of unilabel and multilabel classification, the data balancing approaches commonly used for unilabel classification are not suitable for multilabel classification [21]. In unilabel classification, balancing techniques, such as oversampling or undersampling, usually focus on ensuring a balanced class distribution for every single label. However, in multilabel classification, each sample can have more than one label, so balancing must consider the combination of existing labels. If only traditional balancing methods are used, the risk of integration between labels can be higher because the focus is no longer on just one label per sample but on multiple labels that may have different distributions [22]. For example, a text (named text1) is classified into class labels A and B simultaneously. When undersampling class A (majority), if text1 is removed, it will not only reduce the number of members in class A but also reduce the number of members in class B. Currently, few studies are addressing this matter. This study formulates an undersampling technique that accounts for the interplay of label classes in multilabel scenarios. The dataset chosen for undersampling exhibits variations to enhance classification performance and efficiency during the training of the multilabel classification model.

Numerous studies have emphasized the significance of accuracy, as it reflects the model's proficiency in predicting classification results; however, another crucial aspect is the speed of the training process. Improving efficiency when training classification models becomes especially important in development and research contexts. Slow and resource-intensive training processes not only hinder productivity but can also be a significant financial burden, especially when using expensive computing infrastructure [23]. An approach to increase efficiency is to use techniques such as undersampling in datasets that are considered unimportant or can be omitted [10, 24]. By reducing the number of samples in a data set, the computation time can be significantly reduced while retaining important information. This not only saves time and costs but also allows for greater focus on model development and faster experimentation.

This research aimed to address three hypotheses:

- H1: Adjustments are required to the standard undersampling technique to accommodate differences in multilabel classification datasets.
- H2: Fine-tuning a multilabel classification model necessitates a diverse dataset, as a singular classification dataset is insufficient.
- H3: The dataset produced by the proposed undersampling technique can deliver optimal performance and efficiency.

II. METHODOLOGY

A. Research Object

The research object was reviews of tourist locations on Google Maps. Such reviews contain multiple relevant aspects simultaneously, which makes a multilabel approach ideal. Using multilabel classification, the model can analyze and group different aspects in a review in parallel, providing more comprehensive and in-depth results. Therefore, a multi-label approach is suitable for the case of tourist site reviews because it allows for the assessment of multiple dimensions from a single review. Table I shows some data samples.

B. Research Process

Figure 1 shows the research flow. The initial step was to collect text reviews from Google Maps. 184 points of interest in the Yogyakarta Special Region, Indonesia, were chosen. The text review dataset encompassed a review date range of November 2011 to March 2024. A total of 421,300 reviews were collected. Subsequently, the review text data was filtered based on two criteria: removing review texts containing fewer than 10 words and review texts that did not correspond to any of the designated label classes. After filtering, the data included 157,574 records.

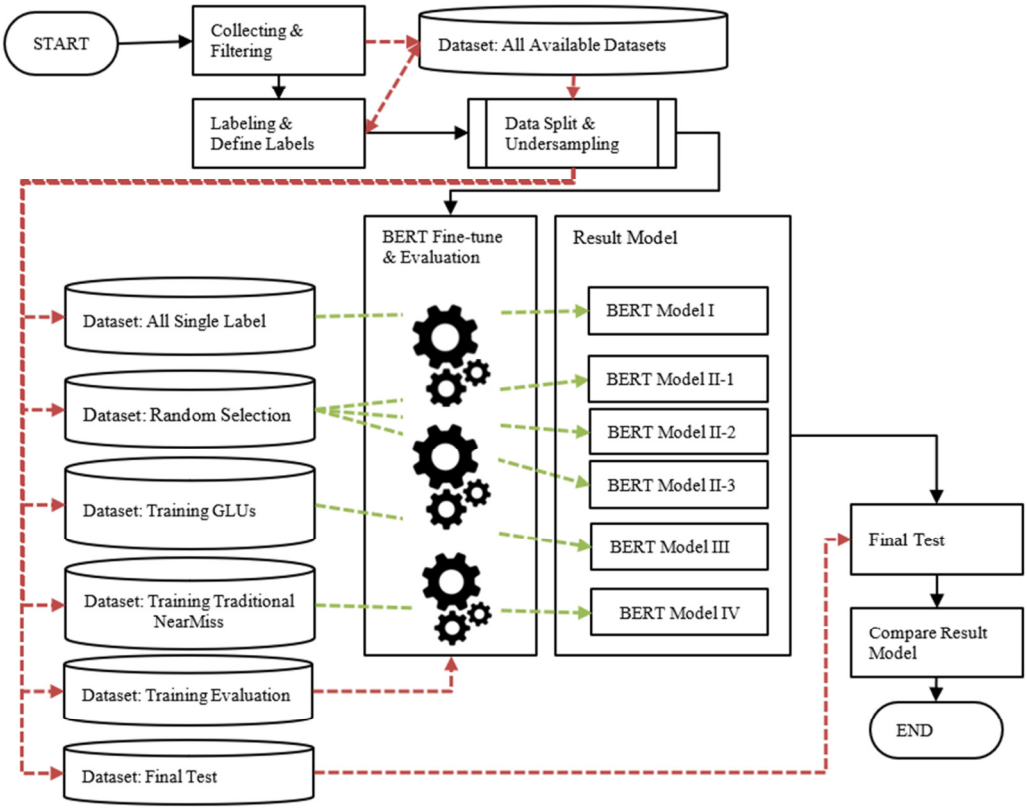

Fig. 1. Process of assessing multilabel classification models.

TABLE I. SAMPLES FROM THE DATASET

No	Text review (Indonesian)	Text review (English)	Class example
1	Tempatnya bagus dan tiket masuk hanya 4k/orng sma parkir 3k	The place is good and the entrance ticket is only 4k/person and parking is 3k	Price
2	Lokasi mudah dijangkau walaupun masuk, tiket gratis, parkir luas	Location is easy to reach even when entering, free tickets, spacious parking	Access Price Facility
3	Parkirannya luas dan teduh, bawa mobil atau motor aman. Aksec masuknya saat ini msh terkendala bbrp spot proyek bandara. Hati2 saja	The parking lot is spacious and shady, it's safe to bring a car or motorbike. Access to entry is currently still hampered by several airport project spots. Just be careful	Facility Access
4	Di Pantai Parangtritis ini kita bisa menyewa motor trail, ada ATV, ada jeep, dan ada delman juga. Kebetulan pas ke sini naik delman, biayanya 50k untuk diantar ke satu ujung saja dan 100k kalau mau dari satu ujung ke ujung lainnya. Di sini juga banyak yang menyewakan tikar. Ada yang di pinggir pantai dan ada juga yang di bawah rindangnya pepohonan.	At Parangtritis Beach, we can rent trail motorbikes, ATVs, jeeps, and wagons too. Coincidentally, when I came here by carriage, it cost 50k to take it to one end only and 100k to go from one end to the other. Many people rent mats here. Some are on the beach and some are under the shade of trees.	Activity Price Facility Weather

The labeling process was then performed for each review text, along with the definition of the label classes. Each review text can be classified into several pre-determined class labels. 11 class labels were determined, namely facilities, services, access, culinary, photo spots, views, prices, visitors, weather, family, and activities (Table II). The labeling process was carried out manually using expert judgment. 20 individuals were employed to carry out the manual labeling process. To mitigate differences in manual classification between individuals, the researchers conducted random inspections. In step 2, each class label was converted to combined labels in binary form as in Table II.

TABLE II. LIST OF LABELS AT EXPERIMENTS

No	Labels in Indonesian	Labels in English	Description
1	Fasilitas	Facility	Facilities offered by tourist locations, including parking facility
2	Pelayanan	Services	Services provided by the tourist location manager
3	Akses	Access	Road access to tourist locations, including distance from the center of the district
4	Kuliner	Culinary	Food and beverages offered in tourist locations.
5	Spot_foto	Photo_spot	Photo spots offered in the tourist locations.
6	Pemandangan	Views	View of the tourist area
7	Harga	Price	Prices of products and services at tourist locations
8	Pengunjung	Visitors	Comments regarding visitors to tourist locations
9	Cuaca	Weather	The weather in the tourist area including fog, hot air, and cold air.
10	Keluarga	Family	Comments on the comfort of traveling with family
11	Aktivitas	Activity	Comments regarding activities that can be done at tourist locations.

TABLE III. DATASET DETAILS

Dataset No.	Dataset	Description	Data quantity	Usage	Result model number
I	All Single label	No Undersampling; random selection All review text with single/1 labels	39,949	Training dataset	I
II-1	Random Selection 1	No Undersampling; random selection All review text in similar proportion (3 times)	40,296	Training dataset	II-1
II-2	Random Selection 2		42,556		II-2
II-3	Random Selection 3		39,615		II-3
III	Training GLU	GLU	40,979	Training dataset	III
IV	Traditional NearMiss	NearMiss Undersampling applied to multilabel; Using unilabel undersampling for multilabel cases	36,572	Training dataset	IV
V	Training Evaluation	Training Evaluation	41,828	Training evaluation dataset	-
VI	Final Test	Final Test	39,129	Final test dataset	-

The collected and labeled text review dataset was divided into five datasets shown in Table III (Step 3). There are three categories of dataset usage: training, training evaluation, and final test. The training dataset was utilized to refine the multilabel classification model. The five training datasets were used to train the BERT multilabel classification models. The training evaluation and the final test datasets contain distinct records compared to the training dataset to ensure that the model developed with the training dataset applies to various datasets.

Each training dataset (t I, II-1, II-2, II-3, and III) was employed for the fine-tuning of the multilabel classification models (Step 4). The fine-tuning process utilized several parameters, as outlined in Table IX. Each BERT model was subsequently evaluated on the final test dataset (Step 5) using Accuracy, Precision, Recall, and F1-score. Finally, each model was evaluated based on performance and time efficiency.

C. Group Level Undersampling Method

A Group Level Undersampling (GLU) technique was devised to handle the undersampling of multilabel classification training data. GLU is an improvement of the existing unilabel undersampling method, NearMiss. The most fundamental difference is in the undersampling process, which is carried out at the group level, and the multilabel is converted into combined labels. Algorithm 2 describes the GLU method. Two libraries are needed: NearMiss from imblearn.under_sampling and TfidfVectorizer from sklearn.feature_extraction.text. For example, the 18 sentences in Table IV were processed using GLU.

```
1  Algorithm custom-undersampling
2  input: imbalanced-multilabel-dataset D
3  output: resampledDatasets RD

4  import libraries
5  from imblearn.under_sampling import NearMiss
6  from sklearn.feature_extraction.text import
   TfidfVectorizer
7  resampledDatasets ← {}
8  SD ← sortToLeveledDataset(D)
9  for i from 1 to len(SD) do
10 ratio ← sum(SD[i].values())/len(SD[i])
11 if ratio > ratioBaseLine then
12 combinedLabelsLevelN ← list(SD.keys())
13 bottomBaseLine ← smallest(SD.values())
14 upperBaseLine ← round(bottomBaseline * 120%)
15 if upperBaseLine < minMember then
16 upperBaseLine ← minMember
17 isTooLow← True
18 classToProcess ← any Labels with member
   more than upperBaseLine
19 textsLevelN ← list of texts in group level
   N (from D)
20 indexCombinedLabelsLevelN ← list of labels
   in group level N (from D)
21 j ← 0
22 while classToProcess is not empty do
23 countOldData ← len(textsLevelN)
24 vectorizer ← TfidfVectorizer()
25 X ←
   vectorizer.fit_transform(textsLevelN)
26 dictSamplingStrategy = {}
27 for k from 0 to len(classToProcess) do
28 dictSamplingStrategy[classToProcess[k]]
   ← upperBaseLine
   nm ←
29 NearMiss(sampling_strategy=
   dictSamplingStrategy, version=2)
30 nm.fit_resample(X,
   indexCombinedLabelsLevelN)
31 sample_indices ← nm.sample_indices_
32 textsLevelN_resampled ← copy all texts
   whose index is in sample_indices
   indexCombinedLabelsLevelN_resampled ←
33 copy all labels whose index is in
   sample_indices
34 countNewData ←
   len(textsLevelN_resampled)
35 if countOldData == countNewData then
36 exit loop
37 increment j by 1
38 return textsLevelN_resampled,
   indexCombinedLabelsLevelN_resampled
```

The first phase of the GLU technique involves converting multilabels into combined labels and categorizing them into certain levels according to the number of labels (pseudo-code line 8). Combined labels are formed by sorting single labels, then giving binary values, and then combining them into a string. For example, D1 is classified into the labels "sound", and not classified into the labels "drink" and "eat". So the combined labels formed are "001". Furthermore, as there is only one single label that has a True value, its group level is 1.

To maintain the consistency of combined labels, the order of single labels is "minum/drink", "makan/eat", and "suara/sound".

TABLE IV. EXAMPLE DATASET - COMBINED LABELS AND GROUPED TO LEVEL

Doc. no	Sentences	Labels			Combined labels	Group level
		drink	eat	sound		
D1	dog barks in the morning	0	0	1	001	1
D2	cat meows at home	0	0	1	001	1
D3	cow moos in the field	0	0	1	001	1
D4	cow moos and the dog barks	0	0	1	001	1
D5	bone is eaten by the dog	0	1	0	010	1
D6	cat eats cat food	0	1	0	010	1
D7	dog eats meat	0	1	0	010	1
D8	cow drinks milk	1	0	0	100	1
D9	dog drinks water	1	0	0	100	1
D10	cow moos and dog eats and cat meows	0	1	1	011	2
D11	dog does not meow but the cat eats	0	1	1	011	2
D12	cow moos, the cat eats	0	1	1	011	2
D13	dog barks and the cat eats	0	1	1	011	2
D14	cow drinks water and eats grass	1	1	0	110	2
D15	dog drinks milk and the cat eats meat	1	1	0	110	2
D16	dog and the cat drink water and the cow moos	1	0	1	101	2
D17	cow eats grass, dog drinks, and cat meows	1	1	1	111	3
D18	cat eats, dog barks, and drinks water	1	1	1	111	3

Upon the creation of the combined labels, the grouped level is chosen for undersampling processing. The choosing of grouped levels continues from group level 1 until group level 3 for this example, and group level 11 for the actual experiment (pseudo-code line 9). Next, the ratio is calculated between the number of documents and the number of combined labels in the level-*n* group (1). *ratioBaseLine* is the limit parameter at which the undersampling process on the *n*-level group will be carried out. In this example, *ratioBaseLine* was set to 2. If the ratio is more than 2, then group level-*n* will not be undersampled.

$$ratio = \frac{number\ of\ text\ reviews\ in\ a\ grouped\ level-n}{list\ of\ combined\ labels\ at\ level-n} \quad (1)$$

Lines 13-17 are used to define the target value of undersampling on each level-*n* group. To calculate the target number of members or *upperBaseLine*, the *bottomBaseLine* value or the minimum number of members on level-*n* is needed. Variable *upperBaseLine* is calculated based on *bottomBaseLine*+20%. If *bottomBaseLine* is less than the *minMember* variable, which is the minimum target number of members on each level-*n* group, then *upperBaseLine* is set to *minMember*. The *minTarget* variable aims to maintain if there is a number of members on the combined labels that is 0 or too small. The value of the combined labels' members that is too small can make the *upperBaseLine* value or the target number of members too small, and many training samples are deleted. After the target number of members is determined, a dictionary is created, which is used for the undersampling parameters of the NearMiss V2 function (line 18).

The next step is to convert the document text into a TF-IDF vector using the *TfidfVectorizer* function (lines 24-25). For example, to convert the first sentence "dog barks in the morning" into a TF-IDF vector, first, the Term Frequency (TF) for each word in the sentence is calculated. Next, the Inverse Document Frequency (IDF) is calculated for each word. IDF measures how important a word is in the entire set of documents. The TF-IDF value is calculated by multiplying the TF and IDF for each word, and T a TF-IDF vector is obtained that represents the importance of each word in the sentence relative to the entire set of documents.

Next, lines 29-31 are the undersampling process using the NearMiss V2 algorithm. NearMiss v2 has several advantages compared to NearMiss v1 and v3 [25, 26]. One of the main advantages of NearMiss v2 is its ability to select a majority sample that has the closest average distance to the three farthest minority samples. This allows NearMiss v2 to be more effective in maintaining a better representation of the minority class compared to NearMiss v1, which only considers the average distance closest to the nearest minority sample. In addition, NearMiss v2 is also superior to NearMiss v3, which selects the majority samples based on the closest average distance to several nearest minority neighbors, because NearMiss v2 focuses more on truly relevant and representative samples. NearMiss v2 was selected for GLU because it provides better balance in the resulting dataset. After the undersampling process, the number of members after undersampling will be compared with the number of members before undersampling for the level-*n* group (lines 34-36). If there is no change in the number of members, the undersampling process for the level-*n* group will be considered complete, and the process will continue to the next level group. In the example dataset group level 1, there are 3 combined labels and 9 text documents in that class, namely D1-D9. First, the distance between the text document in CL 001 and the text document in other CLs (010 and 100) is calculated. Next, the average distance for each document text is calculated. The results of distance calculations for text documents CL 001 and CL 010 can be seen in Tables V and VI. As the undersampling target for this group level is 2 members, the 2 members with the smallest average are looked for as samples, and the others are removed from the collection. In CL 001, the sample document texts selected are D1 and D4. Meanwhile, in CL 010, the sample document texts selected are D5 and D7. In CL 100, the number of members is 2, so sample data deletion/selection is not processed. The GLU process finishes when all group levels are processed. The final statistics of the GLUs process for the example dataset can be seen in Table VII. Figure 3 shows an illustration of the actual GLU process for the example dataset in Table IV. The static parameters used in the GLU process for the example dataset and actual experiment can be seen in Table VIII.

TABLE V. DISTANCE OF DOCS USING COSINE SIMILARITY FOR EXAMPLE DATASET GROUP LEVEL 1 – CL 001

Text document		Combined labels	D5	D6	D7	D8	D9	Average
			Bone is eaten by the dog	Cat eats cat food	Dog eats meat	Cow drinks milk	Dog drinks water	
D1	dog barks in the morning	001	1.2693	1.4142	1.3181	1.4142	1.3181	1.34678
D2	cat meows at home	001	1.4142	1.1432	1.4142	1.4142	1.4142	1.36
D3	cow moos in the field	001	1.3379	1.4142	1.4142	1.2686	1.4142	1.36982
D4	cow moos and the dog barks	001	1.2808	1.4142	1.3255	1.2757	1.3255	1.32434

TABLE VI. DISTANCE EACH DOCS USING COSINE SIMILARITY FOR EXAMPLE DATASET GROUP LEVEL 1 – CL 010

Text document		Combined labels	D1	D2	D3	D4	D8	D9	Average
			Dog barks in the morning	Cat meows at home	Cow moos in the field	Cow moos and the dog barks	Cow drinks milk	Dog drinks water	
D5	bone is eaten by the dog	010	1.2693	1.4142	1.3379	1.2808	1.4142	1.3361	1.3420
D6	cat eats cat food	010	1.4142	1.1432	1.4142	1.4142	1.4142	1.4142	1.3690
D7	dog eats meat	010	1.3181	1.4142	1.4142	1.3255	1.4142	1.2933	1.3632

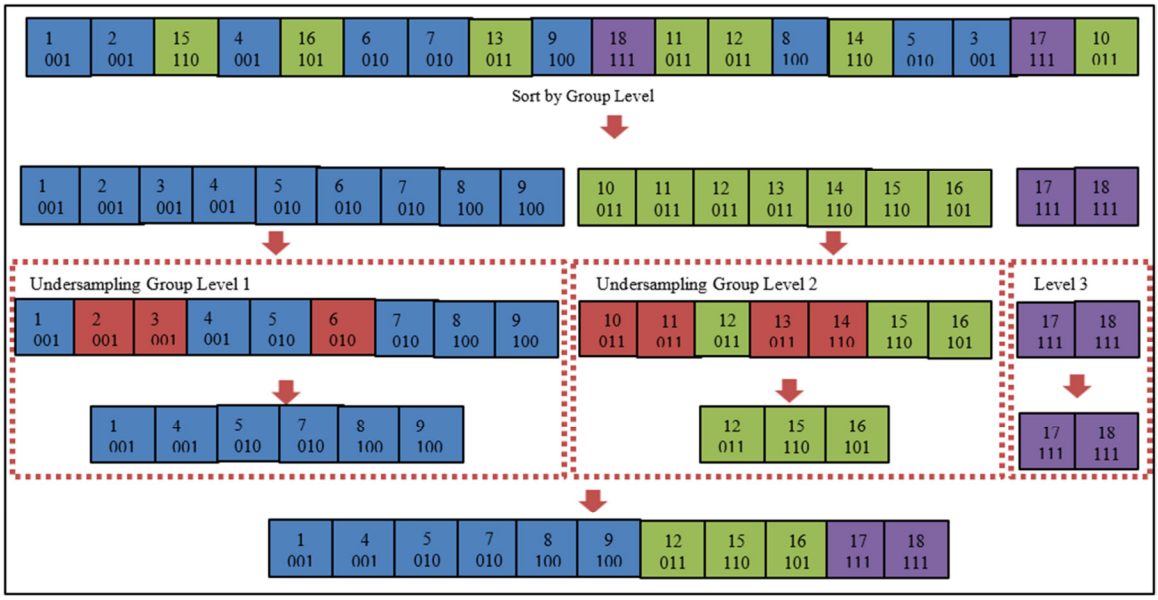

Fig. 2. GLU illustration on the example dataset.

TABLE VII. BEFORE AND AFTER OF EXAMPLE DATASET UNDERSAMPLING PROCESS

BEFORE							
Combined Labels (CL)	001	010	100	011	101	110	111
Members of CL	4	3	2	4	1	2	2
Number of members at the group level	9			7			2
AFTER							
Combined Labels (CL)	001	010	100	011	101	110	111
Member of CL	2	2	2	1	1	1	2
Number of members at the group level	6			3			2

TABLE VIII. STATIC PARAMETERS USED IN GLU

Parameter	Value on example	Value on experiment
ratioBaseLine	2	10
minMember	2	10

D. Fine-Tuning and Model Comparison Method

After all the datasets were built, the study continued with the training or fine-tuning process of the BERT multilabel classification model. BERT is a revolutionary LLM in the field of NLP [27-29]. Developed by Google AI in 2019, BERT can understand the context and relationships between words in a text very well [27]. BERT uses a self-attention architecture to process sequence data, which allows it to capture complex contextual relationships in text [27, 30]. Its main advantage lies in its ability to fine-tune specific tasks (downstream tasks) very efficiently [31]. This is because BERT has been previously trained using text from various sources, so it has extensive knowledge of the language. Thus, the fine-tuning process is highly efficient and does not require excessive time or resources, allowing users to quickly obtain a model that suits their specific needs [32, 33]. Choosing a dataset for a fine-tuning process is very important, considering that the model

will be trained to work on downstream tasks [34, 35]. Currently, there are several high-performance, ready-to-use Indonesian BERT models, namely IndoBERT made by Indobenchmark [36], IndoBERT made by IndoLEM [37], and IndoBERTweet [38]. Each IndoBERT model has its own characteristics based on the training data used. IndoBERT [36] was trained using the Indo4B text corpus, which consists of approximately 23.43 GB of text. Indo4B covers various sources of formal text, such as Wikipedia, news, and a web corpus. IndoBERT [37] was trained using more than 220 million words taken from three main sources: Indonesian Wikipedia (74 million words), news articles from Kompas, Tempo, and Liputan6 (55 million words), and the Indonesian Web Corpus (90 million words). Both IndoBERTs use different training data but focus on formal Indonesian. IndoBERTweet was trained using 409 million word tokens collected from Indonesian tweets for one year, from December 2019 to December 2020 [39]. These training data cover topics in economics, health, education, and government and capture the nuances of everyday language and slang frequently used on Twitter. The main advantage of IndoBERTweet is its ability to understand context and emotions in informal conversations, which makes it more effective in applications that require the understanding of informal and dynamic language. IndoBERTweet was trained using Twitter text (social media), which has characteristics similar to those of the dataset in this study, as both datasets use informal language. The fine-tuning process was carried out using several parameters, as can be seen in Table IX. Each fine-tuning process was carried out on the same computer device using an Intel i7 10700F processor, 16 GB RAM, and NVIDIA GeForce GTX 1660 Super. Experiment 1 was carried out to find the most efficient learning rate and epochs for the selected case. The learning rate and epochs found were used for Experiments 2, 3, and 4, which provide a benchmark for multilabel classification models finetuned by GLU datasets versus random selection datasets.

TABLE IX. FINE-TUNED HYPERPARAMETER COMBINATION

Experiment	Datasets	Batch	Learning rate	Epochs
1	Dataset III	4,8	1e-2, 1e-3, 1e-4, 1e-5	1,3,5
2	Dataset I	4,8	1e-4, 1e-5	5
3	Datasets II-1, II-2, II-3	4,8	1e-4, 1e-5	5
4	Dataset IV	4,8	1e-4, 1e-5	5

E. Group Level Undersampling Result

Tables X and XI show the data statistics before and after the GLU process. As can be seen in Table X, there is no change in the number of combined labels. This demonstrates the variety of information effectively preserved by the GLU method. Based on Table XI, the data changes from group levels 1 to 5 show a significant decrease in the mean (avg) and standard deviation (stdev) after the GLUs process. (exceeding 10) and then undersampled to *minMember*.

In group level 1, the mean value drops from 8218.09 to 2867.73, and the standard deviation drops from 4506.13 to 165.81. This indicates that after the change, the data becomes more consistent and less spread out, with values closer to the new lower mean. This decrease could indicate a reduction in the variability and scale of the data. In group level 2, the mean value drops from 706.89 to 34.89, and the standard deviation drops from 455.52 to 0.80.

TABLE X. BEFORE AND AFTER THE DATASET UNDERSAMPLING PROCESS

		Before	After	
			Traditional NearMiss	GLU
Number of CLs		1,390	947	1,390
Members of CL	1	11	11	11
	2	55	55	55
	3	164	163	164
	4	317	265	317
	5	368	228	368
	6	273	133	273
	7	138	64	138
	8	47	23	47
	9	16	5	16
	10	1	0	1
	11	0	0	0
Number of members at group level	1	90,399	20,915	31,545
	2	38,879	8586	1,919
	3	16,690	3986	1,606
	4	7,150	1853	2,454
	5	2,889	777	1,886
	6	1,108	320	1,108
	7	347	96	347
	8	86	32	86
	9	25	6	25
	10	2	0	2
	11	0	0	0

TABLE XI. GLUS VS TRADITIONAL NEARMISS RESULTS DATA STATISTICS

Group Level	Before				Traditional NearMiss				GLUs			
	Min	Max	Avg	Stdev	Min	Max	Avg	Stdev	Min	Max	Avg	Stdev
1	2461	17410	8218	4506	541	2779	1901.36	628.27	2461	2953	2867.72	165.81
2	29	1716	706.9	455.5	23	644	156.11	144.29	29	35	34.89	0.80
3	2	409	101.8	89.68	1	142	24.45	23.95	2	10	2.13	0.78
4	1	147	22.56	24.62	1	50	6.99	8.17	1	10	9.79	1.08
5	1	56	7.85	9.43	1	24	3.41	2.54	1	10	7.74	3.10
6	1	32	4.06	4.74	1	15	2.41	2.25	1	32	4.06	4.73
7	1	19	2.51	2.75	1	8	1.5	1.15	1	19	2.51	2.75
8	1	7	1.83	1.48	1	3	1.39	0.71	1	7	1.83	1.48
9	1	3	1.56	0.79	1	2	1.2	0.4	1	3	1.56	0.79
10	0	2	-	0	0	-	-	-	2	-	-	-
11	0	0	-	0	0	-	-	-	0	-	-	-

This drastic decrease indicates that after the change, the data becomes much more homogeneous with values that are very close to each other and a much lower mean. Group levels 3 to 5 also show a similar downward trend. In group level 3, the average value drops from 101.77 to 9.79, and the standard deviation drops from 87.68 to 1.08. In group level 4, the average value drops from 22.56 to 7.74, and the standard deviation drops from 24.62 to 3.12. In group level 5, the average value drops from 7.85 to 5.13, and the standard deviation drops from 9.43 to 3.46. There is a condition where the max value or the largest number of members is 10. The value 10 comes from the *minMember* parameter in the GLU process. This shows that there is a very large difference in the number of members

III. RESULTS AND DISCUSSION

Table XI shows the results of traditional NearMiss and GLU undersampling. The number of CLs in traditional NearMiss shows that there is a change from 1,390 to 947, which means that 443 CLs were lost after the traditional NearMiss undersampling process. The greater the number of CLs, the more feature information is available. With 443 CLs lost, it means that there are 443 unique feature data lost. When a text is classified into label classes A and B, it will provide different information from label classes C and D. This shows that traditional NearMiss cannot maintain the diversity of information, which is an advantage of GLU.

In group levels 6 to 9, there is no change before and after the GLU process. This is because there is a calculated ratio between the number of documents in the level-*n* group and the number of combined labels in the level-*n* group [algorithm line 9 and (1)]. If the ratio value is less than the *ratioBaseLine* parameter, then the GLU process is not continued for the level-*n* group, and is continued to the next level group. This aims to avoid reducing the dataset, which is already small in the level-*n* group. However, a weakness emerged during testing, as in groups level 6 and 7, the GLU process was not carried out because the group level ratio exceeded the *ratioBaseLine* parameter value. This ratio limitation aims to ensure that the level-*n* group that has little text data is not processed for undersampling. In group levels 6 and 7, there is one combined label that has 32 and 19 members. However, the ratio in the level-*n* group does not exceed the *ratioBaseLine* value, so the GLU process is not continued for the level-n group, and the dataset results at group levels 6 and 7 are not as good as others.

Experiment 1 addresses H1, as the GLU method can preserve data set variations for multi-label classification. The GLU method still needs to fix the problems in the ratio calculation so that there are no imbalances such as those in group levels 6 and 7.

A. Model Comparison

As some fine-tuned model results experience underfitting or overfitting, they cannot be used generally. In Experiment 1, all models were trained with learning rates of 1e-2 and 1e-3, and a model with a learning rate of 1e-4 experienced underfitting/ overfitting. The results of fine-tuned models in Experiment 1 that were successful or did not experience underfitting/ overfitting can be seen in Table XII. Results no. 4, 5, 8, and 11 were selected because they had the highest accuracy and F1 score in the final test results. Three fine-tuned models were produced with 5 epochs and one with 3 epochs. The accuracy values of the four best-fine-tuned models had a range of 0.9356-0.9400 and an F1 score range of 0.9462-0.9474. The parameters of Experiments 2 and 3 were learning rates of 1e-4 and 1e-5 with 5 epochs. The learning rates 1e-2 and 1e-3, and epochs 1 and 3 were excluded from Experiments 2 and 3 due to their resultant low accuracy and F1 scores in the final test, which were inferior to the chosen learning rate. Examining underfitting or overfitting, the GLU dataset (dataset III) gave better results than datasets I or II. However, the GLU method still cannot be said to fully overcome underfitting/overfitting when training the model.

TABLE XII. EXPERIMENT 1: FINE-TUNE USING GLUS DATASET (DATASET III)

No	Dataset	Batch	Learning rate	Epochs	Training time	Training Results				Final Test Result	
						Training loss	Eval. loss	Eval. accuracy	Eval. F1	Test accuracy	Test F1
1	Dataset III	4	1E-04	3	15,880.96	0.3489	0.0509	0.8801	0.9719	0.8801	0.9688
2	Dataset III	4	1E-04	5	26,483.76	0.4400	0.0892	0.7984	0.9508	0.7881	0.9430
3	Dataset III	4	1E-05	1	4,986.97	0.2710	0.0386	0.9138	0.9413	0.9099	0.9395
4	Dataset III	4	1E-05	3	14,412.14	0.2255	0.0216	0.9324	0.9458	**0.9356**	**0.9462**
5	Dataset III	4	1E-05	5	27,904.70	0.2152	0.0182	0.9379	0.9472	**0.9400**	**0.9474**
6	Dataset III	8	1E-04	1	4,363.69	0.2319	0.0294	0.9198	0.9431	0.9180	0.9419
7	Dataset III	8	1E-04	3	22,941.80	0.2166	0.0204	0.9348	0.9465	0.9356	0.9465
8	Dataset III	8	1E-04	5	36,611.29	0.2150	0.0206	0.9345	0.9465	**0.9370**	**0.9468**
9	Dataset III	8	1E-05	1	6,915.93	0.2986	0.0596	0.8948	0.9368	0.8907	0.9344
10	Dataset III	8	1E-05	3	18,725.01	0.2376	0.0242	0.9311	0.9455	0.9332	0.9456
11	Dataset III	8	1E-05	5	31,986.32	0.2231	0.0198	0.9362	0.9467	**0.9359**	**0.9463**

Table XIII displays the results of Experiments 2, 3, and 4. At a learning rate of 1e-4, only one model, specifically model 18, does not exhibit under- or overfitting. The two fine-tuned models (12 and 13) trained using the dataset I have accuracy values in the range 0.3038-0.3042 and F1 values in the range 0.6374-0.6483. This happens because the model performs well in identifying some positive labels in a multilabel dataset, but performs poorly in predicting negative labels. This model has difficulty in correctly predicting all classes in a multilabel dataset because it was only trained using text reviews with a single label (dataset I). In the case of multilabel classification, training is needed using a multilabel dataset to be able to predict multilabel classes well. Based on these results, the GLU method handled this problem by using the concept of undersampling based on grouped levels and combined labels. This is proven by the diversity of information that was

successfully maintained after the GLU process. In the results of Experiment 3 (14-20), the three best-fine-tuned models had a final test accuracy range of 0.8959-0.9252 and an F1 score range of 0.9361-0.9436. In Experiment 4 (21 and 22), traditional NearMiss had a final test accuracy range of 0.8578-0.8797 and F1 values in a range of 0.9704-0.9292. In this case, the traditional NearMiss performed below the model using the random method. Based on the data analysis, much multilabel feature information was removed by the traditional NearMiss, which is indicated by the reduction in CL at levels 4-9 (Table X). Experiments 2, 3, and 4 answer H2, as the training of a multilabel classification model requires a varied dataset, and one dataset containing single labels cannot improve the performance of a fine-tuned model.

TABLE XIII. EXPERIMENTS 2, 3, 4: FINE-TUNE BENCHMARK (DATASET I & II & IV)

No	Dataset	Batch	Learning rate	Epoch	Training time	Training Results				Final Test Results	
						Training loss	Eval. loss	Eval. accuracy	Eval. F1	Test accuracy	Test F1
12	Dataset I	4	1E-05	5	13,767.01	0.0063	0.7338	0.2848	0.6018	0.3042	0.6483
13	Dataset I	8	1E-05	5	28,531.75	0.0093	0.5638	0.2844	0.5854	0.3038	0.6374
14	Dataset II-1	4	1E-05	5	13,255.00	0.2088	0.0268	0.9266	0.9444	**0.9252**	**0.9436**
15	Dataset II-1	8	1E-05	5	32,585.74	0.2125	0.0360	0.9062	0.9398	0.9094	0.9397
16	Dataset II-2	4	1E-05	5	12,396.15	0.2072	0.0423	0.8925	0.9365	**0.8959**	**0.9361**
17	Dataset II-2	8	1E-05	5	37,221.92	0.2106	0.0570	0.8684	0.9308	0.8779	0.9314
18	Dataset II-3	8	1E-04	5	7,765.56	0.0256	0.2975	0.5211	0.8095	0.5490	0.7330
19	Dataset II-3	4	1E-05	5	12,215.25	0.2075	0.0380	0.8975	0.9377	**0.8979**	**0.9366**
20	Dataset II-3	8	1E-05	5	14,644.57	0.2107	0.0434	0.8895	0.9359	0.8929	0.9353
21	Dataset IV	4	1E-05	5	10,253.85	0.2065	0.0490	0.8867	0.9323	**0.8697**	**0.9292**
22	Dataset IV	8	1E-05	5	28,619.25	0.2158	0.0542	0.8347	0.9099	0.8478	0.9074

The next step was to examine whether the fine-tuned model trained using the dataset resulting from the proposed GLU method (dataset III) provides better scores than random selection (dataset II) and traditional NearMiss (dataset IV). The four best-fine-tuned models trained using dataset III were compared with the four best-fine-tuned models trained using datasets II and IV. Figure 3 shows the fine-tuning process using datasets II and III, indicating that models 4, 5, 8, and 11 (straight lines) were the best fine-tuned models using training data from dataset III. Models 14, 16, and 19 (dashed lines) were the best fine-tuned models using training data from dataset II. In general, the models trained on dataset III were better than those trained on dataset II. This is shown by the accuracy and F1 scores produced during the fine-tuning process of the model with dataset I, which is higher than the model with dataset III. Therefore, it can be concluded that dataset III, which was produced with the GLU method, offered better results compared to using random selection (dataset II).

Figure 3 also shows that since the first epoch, the models trained with the dataset resulting from the GLU method (dataset III - straight line) achieved quite high values, with accuracy starting from ~0.92 and F1 Score starting from ~0.94. This shows that the proposed GLU method can increase efficiency because the fine-tuned process can be carried out with minimal epochs. This fine-tuning comparison addresses H3: Utilizing a suitable dataset improves the fine-tuning process, resulting in optimal outcomes with fewer epochs. However, the testing remains confined to a comparison of datasets generated by GLU and random selection in the context of fine-tuning multilabel classification.

T-tests were used to evaluate the accuracy and F1 scores based on the final test results, as shown in Table XIV. The results show that there is a significant difference between the models. Based on the accuracy of the final test results, a T-statistic value of 2.098 and a P-value of 0.0488 were obtained, which means that the difference in accuracy is statistically significant.

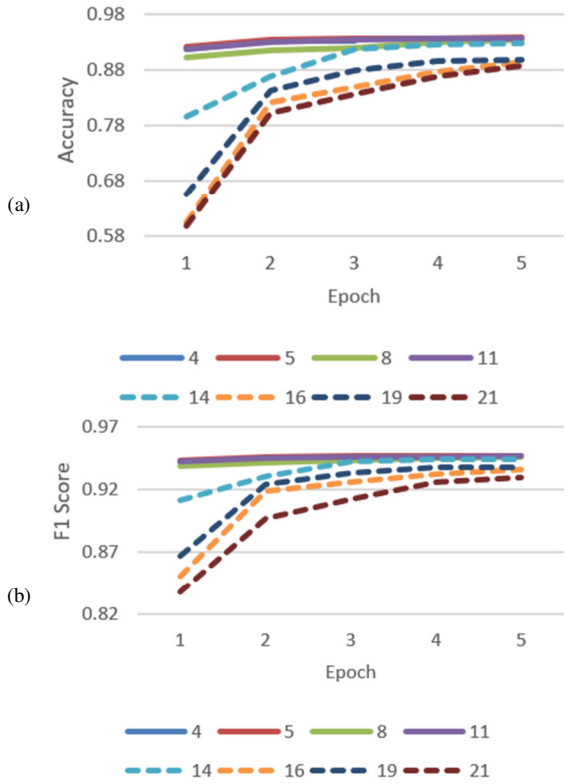

Fig. 3. Fine-tuned process results using datasets II and III.

A large T-statistic value indicates that the difference in average accuracy between the two models is quite strong and is unlikely to occur by chance. The T-test on F1 scores of the final test results revealed a T-statistic value of 2.2557 and a P-value of 0.354, indicating that the difference in F1 scores between the models is statistically significant. A large T-statistic value in the context of the F1 score indicates that the difference in average F1 scores between the two models is

quite significant. Thus, based on accuracy and F1 scores, the models had significantly different performances. The T-statistic and P-value values obtained from these two metrics provide strong evidence that the difference in performance between the models did not occur by chance but is a consistent and reliable result.

TABLE XIV. T-STATISTICS & P-VALUE

	T-statistic	P-value	Conclusion
Accuracy	2.0980	0.0488	Significant
F1	2.2557	0.0354	Significant

IV. CONCLUSION

This study introduced a GLU technique that offers a promising solution for addressing the challenges of imbalanced datasets in multilabel classification (H1). GLU differs from existing undersampling methods in how it handles undersampling for multilabel classification. By converting multilabels into combined labels and categorizing them into group levels, GLU preserves the diversity of combined labels. In these experiments, GLU showed a weakness, as in groups level 6 and 7, the combined labels with 32 and 19 members did not exceed the *ratioBaseLine* value, resulting in halting the GLU process and reducing the quality of the dataset at these levels compared to others. Further research is needed to conduct a sensitivity analysis on *ratioBaseLine* values and examine how the performance changes.

GLU uses group-level and combined-label approaches to improve the NearMiss undersampling method. This approach supports the formation of the dataset needed to optimally train multilabel classification, namely the diversity of feature information combinations in the training data (H2). This experiment contributes knowledge in the form of confirming the theory that to build a multilabel classification model, a dataset is needed that has diverse feature information, in this case including many combined labels.

In addition, the dataset formed from GLUs can provide better performance than traditional NearMiss and random undersampling (H3). Experiments carried out using the IndoBERTweet model showed that GLU not only improves model performance in terms of accuracy and F1 scores but also enhances training efficiency. The T-statistic values of 2.0980 and 2.2557 with P-values of 0.0488 and 0.0354 show a statistically significant difference between the accuracy and F1 scores of the final test results. This T-test value provides evidence that the difference in performance between the two models did not occur by chance but is a statistically significant result. Models trained with the GLU dataset consistently outperformed those trained with randomly selected datasets (all final test accuracy and F1 scores from dataset III were higher than dataset II), indicating the robustness of the proposed approach. The GLU method maintained the variety of label combinations, achieved higher accuracy and F1 scores, and allowed efficient fine-tuning with minimal epochs, reducing computational time and resources. Apart from that, this experiment also proves that the dataset for single-label classification is not suitable for developing multilabel classification models.

Future research should explore the application of the proposed GLU technique to other types of datasets and multilabel classification tasks, such as medical, legal, finance, and social media text datasets, and different labels to assess generalizability. Additionally, investigating the integration of other data balancing methods, such as hybrid oversampling and undersampling techniques, could further enhance performance and efficiency.

REFERENCES

[1] K. Kowsari, K. Jafari Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, and D. Brown, "Text Classification Algorithms: A Survey," *Information*, vol. 10, no. 4, Apr. 2019, Art. no. 150, https://doi.org/10.3390/info10040150.

[2] F. Herrera, F. Charte, A. J. Rivera, and M. J. Del Jesus, "Multilabel Classification," in *Multilabel Classification*, Springer International Publishing, 2016, pp. 17–31.

[3] Q. Li *et al.*, "A Survey on Text Classification: From Traditional to Deep Learning," *ACM Transactions on Intelligent Systems and Technology*, vol. 13, no. 2, pp. 1–41, Apr. 2022, https://doi.org/10.1145/3495162.

[4] J. Tanha, Y. Abdi, N. Samadi, N. Razzaghi, and M. Asadpour, "Boosting methods for multi-class imbalanced data classification: an experimental review," *Journal of Big Data*, vol. 7, no. 1, Sep. 2020, Art. no. 70, https://doi.org/10.1186/s40537-020-00349-y.

[5] A. Kumar, P. Makhija, and A. Gupta, "Noisy Text Data: Achilles' Heel of BERT." arXiv, 2020, https://doi.org/10.48550/ARXIV.2003.12932.

[6] H. Hua, X. Li, D. Dou, C. Z. Xu, and J. Luo, "Noise Stability Regularization for Improving BERT Fine-tuning." arXiv, 2021, https://doi.org/10.48550/ARXIV.2107.04835.

[7] H. Zhang, L. Zhang, and Y. Jiang, "Overfitting and Underfitting Analysis for Deep Learning Based End-to-end Communication Systems," in *2019 11th International Conference on Wireless Communications and Signal Processing (WCSP)*, Xi'an, China, Oct. 2019, pp. 1–6, https://doi.org/10.1109/WCSP.2019.8927876.

[8] J. Bilmes, "Underfitting and overfitting in machine learning," *UW ECE course notes*, vol. 5, 2020.

[9] S. Rahamat Basha and J. K. Rani, "A Comparative Approach of Dimensionality Reduction Techniques in Text Classification," *Engineering, Technology & Applied Science Research*, vol. 9, no. 6, pp. 4974–4979, Dec. 2019, https://doi.org/10.48084/etasr.3146.

[10] P. Gong, Y. Ma, C. Li, X. Ma, and S. H. Noh, "Understand Data Preprocessing for Effective End-to-End Training of Deep Neural Networks." arXiv, 2023, https://doi.org/10.48550/ARXIV.2304.08925.

[11] S. Mohammed *et al.*, "The Effects of Data Quality on Machine Learning Performance on Tabular Data," *Information Systems*, vol. 132, Jul. 2025, Art. no. 102549, https://doi.org/10.1016/j.is.2025.102549.

[12] C. Sun, X. Qiu, Y. Xu, and X. Huang, "How to Fine-Tune BERT for Text Classification?" arXiv, 2019, https://doi.org/10.48550/ARXIV.1905.05583.

[13] T. Wongvorachan, S. He, and O. Bulut, "A Comparison of Undersampling, Oversampling, and SMOTE Methods for Dealing with Imbalanced Classification in Educational Data Mining," *Information*, vol. 14, no. 1, Jan. 2023, Art. no. 54, https://doi.org/10.3390/info14010054.

[14] C. Vairetti, J. L. Assadi, and S. Maldonado, "Efficient hybrid oversampling and intelligent undersampling for imbalanced big data classification," *Expert Systems with Applications*, vol. 246, Jul. 2024, Art. no. 123149, https://doi.org/10.1016/j.eswa.2024.123149.

[15] M. S. Shelke, P. R. Deshmukh, and V. K. Shandilya, "A Review on Imbalanced Data Handling Using Undersampling and Oversampling Technique," *International Journal of Recent Trends in Engineering and Research*, vol. 3, no. 4, pp. 444–449, May 2017, https://doi.org/10.23883/IJRTER.2017.3168.0UWXM.

[16] A. Tripathi, R. Chakraborty, and S. K. Kopparapu, "A Novel Adaptive Minority Oversampling Technique for Improved Classification in Data Imbalanced Scenarios," in *2020 25th International Conference on*

Pattern Recognition (ICPR), Milan, Italy, Jan. 2021, pp. 10650–10657, https://doi.org/10.1109/ICPR48806.2021.9413002.

[17] I. Letteri, A. Di Cecco, A. Dyoub, and G. Della Penna, "A Novel Resampling Technique for Imbalanced Dataset Optimization." arXiv, 2020, https://doi.org/10.48550/ARXIV.2012.15231.

[18] S. S. Rawat and A. K. Mishra, "Review of Methods for Handling Class-Imbalanced in Classification Problems." arXiv, 2022, https://doi.org/10.48550/ARXIV.2211.05456.

[19] J. Bogatinovski, L. Todorovski, S. Džeroski, and D. Kocev, "Comprehensive comparative study of multi-label classification methods," *Expert Systems with Applications*, vol. 203, Oct. 2022, Art. no. 117215, https://doi.org/10.1016/j.eswa.2022.117215.

[20] A. Law and A. Ghosh, "Multi-Label Classification Using Binary Tree of Classifiers," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 6, no. 3, pp. 677–689, Jun. 2022, https://doi.org/10.1109/TETCI.2021.3075717.

[21] A. Y. Taha, S. Tiun, A. H. A. Rahman, and A. Sabah, "Multilabel over-sampling and under-sampling with class alignment for imbalanced multilabel text classification," *Journal of Information and Communication Technology*, vol. 20, no. 3, pp. 423–456, Jun. 2021, https://doi.org/10.32890/jict2021.20.3.6.

[22] A. N. Tarekegn, M. Giacobini, and K. Michalak, "A review of methods for imbalanced multi-label classification," *Pattern Recognition*, vol. 118, Oct. 2021, Art. no. 107965, https://doi.org/10.1016/j.patcog.2021.107965.

[23] C. Li *et al.*, "DeepSpeed Data Efficiency: Improving Deep Learning Model Quality and Training Efficiency via Efficient Data Sampling and Routing," in *Proceedings of the AAAI Conference on Artificial Intelligence*, Mar. 2024, vol. 38, pp. 18490–18498, https://doi.org/10.1609/aaai.v38i16.29810.

[24] N. S. Chatterji, S. Haque, and T. Hashimoto, "Undersampling is a Minimax Optimal Robustness Intervention in Nonparametric Classification." arXiv, Jun. 19, 2023, https://doi.org/10.48550/arXiv.2205.13094.

[25] N. M. Mqadi, N. Naicker, and T. Adeliyi, "Solving Misclassification of the Credit Card Imbalance Problem Using Near Miss," *Mathematical Problems in Engineering*, vol. 2021, no. 1, 2021, Art. no. 7194728, https://doi.org/10.1155/2021/7194728.

[26] A. Tanimoto, S. Yamada, T. Takenouchi, M. Sugiyama, and H. Kashima, "Improving imbalanced classification using near-miss instances," *Expert Systems with Applications*, vol. 201, Sep. 2022, Art. no. 117130, https://doi.org/10.1016/j.eswa.2022.117130.

[27] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, MN, USA, Mar. 2019, pp. 4171–4186, https://doi.org/10.18653/v1/N19-1423.

[28] D. Sebastian, H. D. Purnomo, and I. Sembiring, "BERT for Natural Language Processing in Bahasa Indonesia," in *2022 2nd International Conference on Intelligent Cybernetics Technology & Applications (ICICyTA)*, Bandung, Indonesia, Dec. 2022, pp. 204–209, https://doi.org/10.1109/ICICyTA57421.2022.10038230.

[29] C. Jhakal, K. Singal, M. Suri, D. Chaudhary, B. Kumar, and I. Gorton, "Detection of Sexism on Social Media with Multiple Simple Transformers," in *CLEF 2023: Conference and Labs of the Evaluation Forum*, Thessaloniki, Greece, Sep. 2023.

[30] A. Vaswani *et al.*, "Attention Is All You Need." arXiv, 2017, https://doi.org/10.48550/ARXIV.1706.03762.

[31] M. V. Koroteev, "BERT: A Review of Applications in Natural Language Processing and Understanding." arXiv, Mar. 22, 2021, https://doi.org/10.48550/arXiv.2103.11943.

[32] Y. Hao, L. Dong, F. Wei, and K. Xu, "Visualizing and Understanding the Effectiveness of BERT." arXiv, Aug. 15, 2019, https://doi.org/10.48550/arXiv.1908.05620.

[33] Y. Zhou and V. Srikumar, "A Closer Look at How Fine-tuning Changes BERT." arXiv, Mar. 16, 2022, https://doi.org/10.48550/arXiv.2106.14282.

[34] M. Shen, "Rethinking Data Selection for Supervised Fine-Tuning." arXiv, Feb. 08, 2024, https://doi.org/10.48550/arXiv.2402.06094.

[35] F. Kang *et al.*, "Get more for less: Principled Data Selection for Warming Up Fine-Tuning in LLMs." arXiv, May 05, 2024, https://doi.org/10.48550/arXiv.2405.02774.

[36] B. Wilie *et al.*, "IndoNLU: Benchmark and Resources for Evaluating Indonesian Natural Language Understanding." arXiv, Oct. 08, 2020, https://doi.org/10.48550/arXiv.2009.05387.

[37] F. Koto, A. Rahimi, J. H. Lau, and T. Baldwin, "IndoLEM and IndoBERT: A Benchmark Dataset and Pre-trained Language Model for Indonesian NLP." arXiv, Nov. 02, 2020, https://doi.org/10.48550/arXiv.2011.00677.

[38] F. Koto, J. H. Lau, and T. Baldwin, "IndoBERTweet: A Pretrained Language Model for Indonesian Twitter with Effective Domain-Specific Vocabulary Initialization." arXiv, Sep. 10, 2021, https://doi.org/10.48550/arXiv.2109.04607.

[39] N. Sureja, N. Chaudhari, P. Patel, J. Bhatt, T. Desai, and V. Parikh, "Hyper-tuned Swarm Intelligence Machine Learning-based Sentiment Analysis of Social Media," *Engineering, Technology & Applied Science Research*, vol. 14, no. 4, pp. 15415–15421, Aug. 2024, https://doi.org/10.48084/etasr.7818.