A COMPARISON OF IMBALANCED DATA HANDLING METHODS FOR PRE-TRAINED MODEL IN MULTI-LABEL CLASSIFICATION OF STACK OVERFLOW

Miss Arisa Umparat

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science in Statistics
Department of Statistics
FACULTY OF COMMERCE AND ACCOUNTANCY
Chulalongkorn University
Academic Year 2022

การเปรียบเทียบวิธีการจัดการข้อมูลที่ไม่สมดุลสำหรับแบบจำลองที่ได้รับการฝึกฝนแล้วสำหรับวิธีการจำแนกประเภทแบบหลายลาเบลในสแต็กโอเวอร์โฟลว์

น.ส.อริสา อัมภรัตน์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาสถิติ ภาควิชาสถิติ
คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2565

Thesis Title	A COMPARISON OF IMBALANCED DATA HANDLING METHODS FOR PRE-TRAINED MODEL IN MULTI-LABEL CLASSIFICATION OF STACK OVERFLOW
By	Miss Arisa Umparat
Field of Study	Statistics
Thesis Advisor	Assistant Professor SURONAPEE PHOOMVUTHISARN, Ph.D.

Accepted by the FACULTY OF COMMERCE AND ACCOUNTANCY, Chulalongkorn University in Partial Fulfillment of the Requirement for the Master of Science

-- Dean of the FACULTY OF COMMERCE AND ACCOUNTANCY

(Professor Wilert Puriwat, Ph.D.)

THESIS COMMITTEE

-- Chairman

(Professor SEKSAN KIATSUPAIBUL, Ph.D.)

-- Thesis Advisor

(Assistant Professor SURONAPEE PHOOMVUTHISARN, Ph.D.)

-- Examiner

(Assistant Professor PURIPANT RUCHIKACHORN, Ph.D.)

-- External Examiner

(Chalee Thammarat, Ph.D.)

อริสา อัมภรัตน์ : การเปรียบเทียบวิธีการจัดการข้อมูลที่ไม่สมดุลสำหรับแบบจำลองที่ได้รับการฝึกฝนแล้วสำหรับวิธีการจำแนกประเภทแบบหลายลาเบลในสแต็กโอเวอร์โฟลว์. (A COMPARISON OF IMBALANCED DATA HANDLING METHODS FOR PRE-TRAINED MODEL IN MULTI-LABEL CLASSIFICATION OF STACK OVERFLOW) อ.ที่ปรึกษาหลัก : ผศ. ดร.สุรณพีร์ ภูมิวุฒิสาร

การจัดประเภทแท็กมีความสำคัญในสแต็กโอเวอร์โฟลว์ นอกจากจะช่วยให้ผู้ใช้สามารถค้นหาข้อมูลแล้วยังช่วยเสนอวิธีแก้ปัญหาที่เกี่ยวข้องอย่างมีประสิทธิภาพมากขึ้นอีกด้วย เนื่องจากคำถามในโพสต์สามารถมีได้หลายแท็กดังนั้นการจัดประเภทแท็กในสแต็กโอเวอร์โฟลว์จึงถือเป็นเรื่องที่ท้าทาย ซึ่งส่งผลให้เกิดปัญหาความไม่สมดุลระหว่างแท็กกับแท็กทั้งหมด เราจึงนำโมเดลการเรียนรู้เชิงลึกที่ได้รับการฝึกฝนแล้วพร้อมกับชุดข้อมูลขนาดเล็กมาทดลองเพื่อเพิ่มความแม่นยำในการจำแนกหรือการทำนายแท็กได้ โดยใช้เทคนิคการสุ่มตัวอย่างใหม่ที่เหมาะกับการจำแนกประเภทแบบหลายลาเบลโดยเฉพาะ โดยทั่วไปแล้วเพียงแค่ใช้เทคนิคการเรียนรู้ของเครื่องก็สามารถแก้ไขปัญหานี้ได้เช่นกัน แต่มีแค่ไม่กี่งานวิจัยเท่านั้นที่ทดลองว่าเทคนิคการสุ่มตัวอย่างใหม่แบบใดที่สามารถปรับปรุงประสิทธิภาพของโมเดลเชิงลึกโดยใช้แบบจำลองที่ได้รับการฝึกฝนแล้วสำหรับการทำนายแท็ก เพื่อจัดการกับข้อจำกัดนี้ เราได้ทดลองเพื่อประเมินประสิทธิภาพของ ELECTRA ซึ่งเป็นโมเดลการเรียนรู้เชิงลึกที่ได้รับการฝึกฝนแล้วที่ทรงพลัง อีกทั้งยังเสริมด้วยด้วยเทคนิคการสุ่มตัวอย่างใหม่แบบหลายลาเบลเพื่อลดความไม่สมดุลของข้อมูลที่ทำให้เกิดการติดลาเบลผิดในโพสต์ของสแต็กโอเวอร์โฟลว์ เราเปรียบเทียบเทคนิคการสุ่มใหม่ 6 เทคนิค ประกอบไปด้วย ML-ROS, MLSMOTE, MLeNN, MLTL, ML-SOL และ REMEDIAL เพื่อหาวิธีที่ดีที่สุดในการลดความไม่สมดุลของข้อมูล พร้อมทั้งปรับปรุงความแม่นยำในการคาดทำนายแท็ก ซึ่งผลลัพธ์ของเราแสดงให้เห็นว่า MLTL เป็นตัวเลือกที่มีประสิทธิภาพมากที่สุดในการจัดการกับความไม่สมดุลในการจำแนกประเภทหลายลาเบลสำหรับข้อมูลในสแต็กโอเวอร์โฟลว์ในการเรียนรู้เชิงลึก โดยเทคนิค MLTL ทำได้ 0.517, 0.804, 0.467 และ 0.98 จากตัวชี้วัด Precision@1, Recall@5, F1-score@1 และ AUC ตามลำดับ แต่ MLeNN กลับทำได้แค่เพียง 0.323, 0.648, 0.277 และ 0.95 จากตัววัดผลเดียวกัน

สาขาวิชา	สถิติ	ลายมือชื่อนิสิต	
ปีการศึกษา	2565	ลายมือชื่อ อ.ที่ปรึกษาหลัก	

6480507026 : MAJOR STATISTICS

KEYWORD: Pre-trained model tag prediction multi-label classification Stack Overflow imbalanced data deep learning

Arisa Umparat : A COMPARISON OF IMBALANCED DATA HANDLING METHODS FOR PRE-TRAINED MODEL IN MULTI-LABEL CLASSIFICATION OF STACK OVERFLOW. Advisor: Asst. Prof. SURONAPEE PHOOMVUTHISARN, Ph.D.

Tag classification is essential in Stack Overflow. Instead of combining through pages or replies of irrelevant information, users can easily and quickly pinpoint relevant posts and answers using tags. Since User-submitted posts can have multiple tags, classifying tags in Stack Overflow can be challenging. This results in an imbalance problem between labels in the whole labelset. Pretrained deep learning models with small datasets can improve tag classification accuracy. Common multi-label resampling techniques with machine learning classifiers can also fix this issue. Still, few studies have explored which resampling technique can improve the performance of pre-trained deep models for predicting tags. To address this gap, we experimented to evaluate the effectiveness of ELECTRA, a powerful deep learning pre-trained model, with various multi-label resampling techniques in decreasing the imbalance that induces mislabeling in Stack Overflow's tagging posts. We compared six resampling techniques, such as ML-ROS, MLSMOTE, MLeNN, MLTL, ML-SOL, and REMEDIAL, to find the best method to mitigate the imbalance and improve tag prediction accuracy. Our results show that MLTL is the most effective selection to tackle the inequality in multi-label classification for our Stack Overflow data with deep learning scenarios. MLTL achieved 0.517, 0.804, 0.467, and 0.98 from the metrics Precision@1, Recall@5, F1-score@1, and AUC, respectively. Conversely, MLeNN gained only 0.323, 0.648, 0.277, and 0.95 from the same metrics.

Field of Study:	Statistics	Student's Signature	
Academic Year:	2022	Advisor's Signature	

ACKNOWLEDGEMENTS

This thesis is supported by my family, my dedicated advisor who always recommends the best method to get through all the obstacles, and my friends, especially Mr. Peerachai Banyongrakkul and Mr. Passin Pornvoraphat who suggested the solutions and resolved the challenged coding.

Also, Mr. Kantapong Visantavarakul and Ms. Manusaporn Treerungroj for clarifying the essential evaluation metrics, Ms. Pratana Atikhomkamalasai for proofreading and editing, and anonymous reviewers for advising.

Arisa Umparat

TABLE OF CONTENTS

LIST OF TABLES

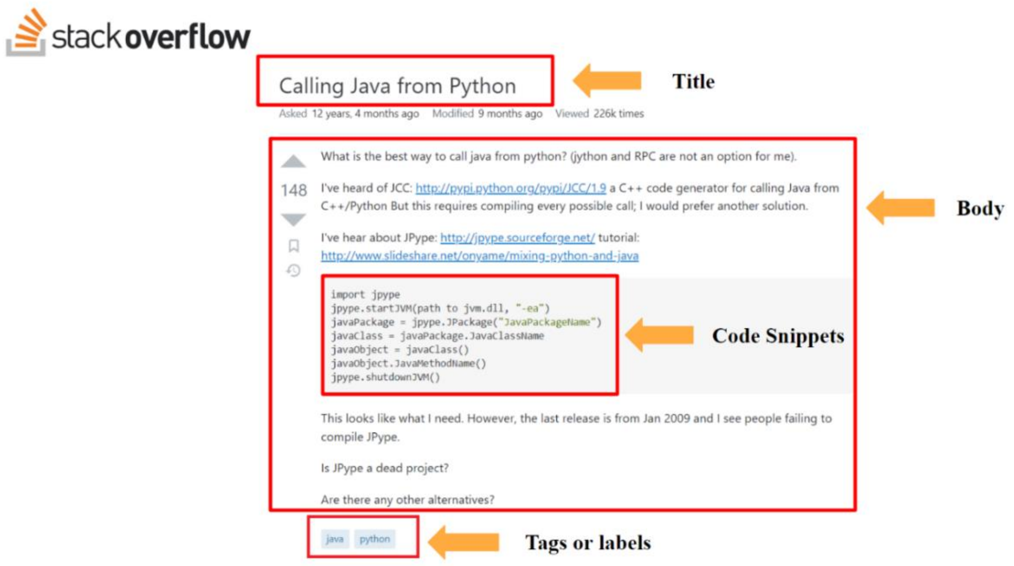
LIST OF FIGURES

Chapter I

INTRODUCTION

1.1 Overview

Coding is a crucial talent for many people today, including professional developers, hobby programmers, and anyone interested in acquiring this technological competency. People frequently look for and follow programming tutorials from online sites to learn and strengthen their coding skills. Stack Overflow, an online Q&A site targeted and visited by programmers worldwide, is one of the most popular resources. The website gives answers to a variety of programming questions and subjects. As seen in Figure 1, Stack Overflow entries have titles, bodies, and tags.



Calling Java from Python

Asked 12 years, 4 months ago Modified 9 months ago Viewed 226k times

148

What is the best way to call java from python? (jython and RPC are not an option for me).

I've heard of JCC: http://pypi.python.org/pypi/JCC/1.9 a C++ code generator for calling Java from C++/Python But this requires compiling every possible call; I would prefer another solution.

I've hear about JPype: http://jpype.sourceforge.net/ tutorial: http://www.slideshare.net/onyame/mixing-python-and-java

```
import jpype
jpype.startJVM(path to jvm.dll, "-ea")
javaPackage = jpype.JPackage("JavaPackageName")
javaClass = javaPackage.JavaClassName
javaObject = javaClass()
javaObject.JavaMethodName()
jpype.shutdownJVM()
```

This looks like what I need. However, the last release is from Jan 2009 and I see people failing to compile JPype.

Is JPype a dead project?

Are there any other alternatives?

java python

Figure 1: An example of Stack Overflow’s question

Still, the tags could be dispersed or mislabeled, resulting in a multi-label categorization challenge (Pant et al., 2018). This is exacerbated by the problem of data imbalance. It is common in multi-label datasets and can cause bias in the classifier (Peng et al., 2021). Data imbalance can have a major impact on model performance in the case of Stack Overflow, which contains a big dataset with thousands of labels. To solve this data imbalance issue and increase tag classification accuracy, proper resampling techniques must be used.

To address data imbalance in multi-label classification, resampling techniques are often utilized. These methods are classified as random or heuristic and include ML-ROS, MLSMOTE, REMEDIAL, MLeNN, MLTL, and MLSOL. Undersampling methods should not be used with MLDs because they can result in a large loss of potentially helpful information during the training phase (Charte, Rivera, Jesús, et al., 2014). LP-ROS is eliminated from our strategy since its algorithm's pseudocode is incomplete, resulting in coding confusion. For severely imbalanced multi-label datasets such as the Stack Overflow dataset, ML-ROS, MLSMOTE, MLeNN, MLTL, ML-SOL, and REMEDIAL are appropriate.

Deep learning-based pre-trained language models have recently improved classification models (Zhou et al., 2019), with BERT serving as a prime example (Charte et al., 2017). However, some BERT variations demand a lot of processing power, rendering them inappropriate for computers with limited resources (Giraldo-Forero et al., 2013). Fortunately, there is a transformer model known as ELECTRA (Clark et al., 2020) that outperforms these versions of BERT on fewer computer resources. ELECTRA develops two transformer models for token replacement detection. When compared to models such as RoBERTa (Liu et al., 2019) and XLNet (Yang et al., 2019), this takes less time and yields higher accuracy on downstream tasks. ELECTRA is a suitable deep learning pre-trained model for handling Stack Overflow-related downstream tasks, such as question answering and text categorization, and is thus a suitable model to test against multi-label resampling strategies.

While ML-ROS and MLSMOTE can tackle imbalanced multi-label problems in deep learning contexts, they have not been examined in the Stack Overflow dataset. Research is scarce on tag recommendations for Stack Overflow utilizing pre-trained transformer-based models. Only pre-trained PTM4Tag has been developed and studied for multi-label classification on the Stack Overflow dataset. The multi-label classification challenge for Stack Overflow data using deep learning scenarios was addressed in this research by modifying several multi-label resampling strategies to deal with imbalanced datasets. To avoid ineffective resampling methods, the data were stratified evenly before applying pre-training ELECTRA, optimized with Adam with a learning rate of 7E-5, to discover the optimal multi-label resampling approach. The performance of several multi-label resampling approaches was assessed using metrics such as Precision@k, Recall@k, F1-score@k, and AUC. MLTL achieved 0.517, 0.804, 0.467, and 0.98 for the metrics Precision@k, Recall@k, F1-score@k, and AUC, respectively.

The paper's contributions include augmenting pre-trained models using various multi-label resampling strategies to handle imbalanced datasets. This had not before been researched or obtained high performance for multi-label classification issues. Based on our findings, the study indicates that MLTL is the optimal multi-label resampling method for the Stack Overflow dataset with deep learning scenarios.

Chapter II
BACKGROUND

2.1 Problem Formulation

A common difficulty in multi-label datasets for classification is data imbalance (Pant et al., 2018), which causes classifier bias (Zhou et al., 2019). The most often utilized resampling techniques to handle this are LP-RUS and LP-ROS developed by Charte et al. (2013). These are examples of multi-label resampling approaches based on LP transformation. However, they are unlikely to solve the imbalance problem. Charte et al. (2015) also published ML-RUS, which aims to remove samples with majority labels, and ML-ROS, which clones samples with minority labels. Due to the co-occurrence of minority and majority labels, some of the minority samples chosen by ML-ROS contain the most common labels. Then, Charte et al. (2015) devised a REMEDIAL technique to address the imbalanced issue by detaching the majority and minority labels. Giraldo-Forero et al. (2013) used SMOTE (Synthetic Minority Over-sampling Technique) in the heuristic oversampling method. Then, Charte et al. (2014) published MLeNN, the first heuristic multi-label undersampling technique. Another approach proposed by Charte et al. (2015) is MLSMOTE, Multilabel Synthetic Minority Oversampling Technique, which uses the instances as seeds to generate new instances. In most multi-label datasets, this is appropriate for several minority labels. Furthermore, Liu et al. (2019) developed MLSOL to investigate imbalance in minority samples based on local characteristics rather than the entire dataset. Recently, Pereira et al. (2020) introduced MLTL to handle the imbalance in the undersampling technique by using the standard Tomek Link algorithm. REMEDIAL-HwR (REMEDIAL Hybridization with Resampling) was proposed by Charte et al. (2019) as three hybrid methods that include several resampling techniques. Liu et al. (2022) recently modified their prior approach by incorporating MLSOL and MLUL, particularly for local label distribution.

Undersampling strategies, whether random or heuristic, should not be used on MLDs since they are not truly unbalanced and result in a considerable loss of potentially relevant information during the training process. LP-ROS is removed from our method since its algorithm's pseudocode is incomplete, which leads to coding misunderstandings. As a result, we choose ML-ROS, REMEDIAL, MLSOL, MLTL, MLeNN, and MLSMOTE because the former removes instances with the most common labelset (i.e. particular combination of label values) while the latter replicates examples with the fewest label sets. These are appropriate for our dataset.

Oversampling methods such as ML-ROS, which deal with individual imbalance evaluations per label rather than whole labelsets, which utilize them to decide which instances will be cloned or eliminated (Charte et al., 2015a), could be useful for our Stack Overflow with many labels and difficult to decide. The oversampling method MLSMOTE creates each minority sample as a seed for a new synthetic sample and is recommended for use with highly imbalanced multi-label datasets such as our Stack Overflow. Furthermore, a study of resampling approaches demonstrated that MLeNN and MLTL are limited by feature designing but not by unambiguous neighbors for large-scale learning (Peng et al., 2021). REMEDIAL and hybridizations do not perform well in deep learning settings because REMEDIAL generates a large number of virtual labelsets and modifies the label space paradigm. ML-ROS and MLSMOTE, on the other hand, are appropriate for resolving imbalanced difficulties in multi-label deep learning. MLSOL, ML-ROS, and MLSMOTE have never been examined for the Stack Overflow dataset, nor have MLTL, MLeNN, and REMEDIAL. LP-ROS is excluded from our approach due to its algorithm's pseudocode is not complete to avoid misunderstanding in coding. We chose ML-ROS, REMEDIAL, MLSOL, MLTL, MLeNN, and MLSMOTE to compare their performance in our large-scale software tag prediction models because of all of the methodologies described.

Pant et al. (2018) spotted the challenges of the multi-label classification problem as removing noisy data from insignificant features in high-dimensionality reduction, degrading data quality from the cleaning process and label dependency, uncertain and imbalance data, label uncertainty from adding unnecessary labels, and drifting labels. However, we only focus on the imbalanced data problem to avoid misclassification due to the classifier's bias towards the labels being increased by the multi-label imbalance.

A deep learning-based technique's pre-trained language model recently improved the categorization model. Devlin et al. (2019) developed the primary BERT. Its variations have been developed, but some of them used high computational techniques (Von der Mosel et al., 2022), making it troublesome and unsuitable for the restricted computing resources (Giraldo-Forero et al., 2013). On the other hand, there is a transformer that outperforms on fewer computational resources. ELECTRA (Clark et al., 2020), is a pre-training task that trains two transformer models for replacement token identification. It is faster and more efficient than RoBERTa (Liu et al., 2019) and XLNet (Yang et al., 2019). When completely trained, it performs better on downstream tasks. As a result, it can help with Stack Overflow-related downstream tasks like Question Answering and Text Classification.

There have not been many studies on tag suggestion combining pre-trained transformer-based models in multi-label classification issue handling with a thousand Stack Overflow labels. He et al. (2022) developed the pre-trained PTM4Tag including BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), ALBERT (Lan et al., 2019), CodeBERT (Feng et al., 2020), and BERTOverflow (Tabassum et al., 2020) specific for multi-label classification, by comparing PTM4Tag with the previous Post2Vec as a baseline model. There has been no previous research referencing pre-trained ELECTRA for tag recommendation in multi-label classification issues from Stack Overflow.

2.2 Research Questions

Which multi-label resampling method is the most effective for dealing with the unbalanced problem in SO posts? By assessing the effectiveness and performance of the pre-trained ELECTRA model, we examine various multi-label resampling approaches, such as ML-ROS, MLSMOTE, MLeNN, MLTL, ML-SOL, and REMEDIAL, to lessen imbalanced datasets from Q&A sites. Precision@k, Recall@k, F1-score@k, and AUC are among the measures we use to evaluate outcomes.

2.3 Objectives

This work attempts to address the issue of imbalanced data by evaluating several multi-label resampling approaches to improve the performance of a deep learning pre-trained model. We concentrate primarily on multi-label classification and the efficiency of ELECTRA, the pre-trained model because ELECTRA can be trained on a single GPU due to our constrained resources. Furthermore, it has never been trained for tag prediction in multi-label classification problems on the Stack Overflow Q&A Site.

Chapter III
LITERATURE REVIEW

3.1 Tag Recommendations

There have been numerous ways created to classify tags for Stack Overflow users. Among them, TagCNN, TagRNN, TagHAN, and TagRCNN are more effective than classic tag recommendation algorithms (Zhou et al., 2019) such as EnTagRec (Wang et al., 2014), TagMulRec (Zhou et al., 2017), and FastTagRec (Liu et al., 2018). Convolutional Neural Networks (CNNs) were recently used as feature extractors in Post2Vec (Xu et al., 2022) with a sigmoid layer instead of the state-of-the-art softmax classification layer. Deep learning models that have been pre-trained have shown the potential in boosting tag categorization accuracy, especially with limited datasets.

3.2 Multi-Label Resampling Methods for Imbalanced Data

The scope of imbalance in traditional classification is determined by comparing the number of instances from the majority and minority classes. Typically, the number of samples from the minority class is significantly lower than that of the majority class (García et al., 2008). To overcome this issue, popular strategies focus on balancing the dataset through undersampling (lowering the size of the majority class), oversampling (raising the size of the minority class), or a mix of both. However, in multi-label classification, the dataset contains hundreds or thousands of labels. As a result, all labels must be considered.

According to the distinction between global and local label density (Kotze, 2022), global label density is the proportion of the total tag of all labels relative to the full rank summation of all labels. Suppose the dataset has 30 label tags with 100 instances and 3 multi-labels, global label density is calculated to be 0.1 or 10%. On the other hand, local label density is determined for each label separately. It considers class distributions by summing the total observations present for the given label and dividing it by the total number of samples in the dataset.

To measure imbalanced data from a thousand labels, L for the total set of labels, L_l for the *l-th* label in this set, and Y_i for the labelset associated with the *i-th* sample in multi-label dataset D. The same labelset can appear in several samples of

D. The threshold is used to cut the set of labels into minority and majority labels. The labels whose $IRLbl$ is larger than $MeanIR$ are called majority labels. On the other hand, the labels whose $IRLbl$ is less than $MeanIR$ are called minority labels (Charte et al., 2015a). But some approaches consider for each label, a label l is defined as a minority when $IRLbl(l)$ is above $MeanIR$ (Charte et al., 2015b). These are the following measures for evaluating the level of imbalance (Charte et al., 2013):

3.2.1 IRperLabel

The ratio between the majority and the considered labels, the higher $IRLbl$ (IRperLabel) could increase the imbalance level.

$$IRLbl(l) = \frac{\underset{l'=Y_1}{\operatorname{argmax}}^{L_{|L|}} \left(\sum_{i=1}^{|D|} h(l',Y_i)\right)}{\sum_{i=1}^{|D|} h(l,Y_i)}, \qquad h(l,Y_i) = \begin{cases} 1 & l \in Y_i \\ 0 & l \notin Y_i \end{cases} \tag{1}$$

3.2.2 MeanIR

The average level of imbalance is estimated as the global imbalance level.

$$MeanIR = \frac{1}{|L|} \sum_{l=L_1}^{L_{|L|}} (IRLbl\,(l)) \tag{2}$$

3.2.3 CVIR

The coefficient of variation of *IRLbl* can indicate if all labels suffer from a similar level of imbalance or if there are huge differences. The higher $CVIR$ induces a larger difference.

$$CVIR = \frac{IRLbl\sigma}{MeanIR}, IRLbl\sigma = \sqrt{\sum_{l=Y1}^{L_{|L|}} \frac{(IRLbl(l) - MeanIR)^2}{|L| - 1}} \tag{3}$$

3.2.4 SCRUMBLE

Recently, there is another measurement named $SCUMBLE$ (Charte, Rivera, del Jesus, et al., 2014), which is based on the Atkinson index and $IRLbl$, that measures the imbalance level of each label instead of the total set of labels, by taking each instance D_i in a multi-label dataset D as a population, and the active labels in D_i as individuals. If the label l is present in instance i then $IRLbl_{il} = IRLbl(l)$. On the contrary, $IRLbl_{il} = 0$. Also, $\overline{IRLbl_l}$ is defined as the average imbalance level of the labels appearing in instance i. The most common is the total number of labels $|L|$. The higher value increases inconsistent frequencies of labels.

$$SCUMBLE\,(D) = \frac{1}{|D|} \sum_{i=1}^{|D|} \left[1 - \frac{1}{\overline{IRLbl_i}} \left(\prod_{l=1}^{|L|} IRLbl_{il}\right)^{1/|L|}\right] \tag{4}$$

To address imbalanced data, numerous resampling strategies are accessible. LP-RUS (Label Powerset Random Undersampling) and LP-ROS (Label Powerset Random Oversampling) are the most used (Charte et al., 2013). LP-RUS removes instances allocated with the most frequent labelset based on LP transformation, as evaluating the entire labelset may not solve the imbalance problem. Later, ML-RUS (Multi-Label Random Undersampling) and ML-ROS (Multi-Label Random Oversampling) (Charte et al., 2015a) are proposed, to delete majority-labeled samples and cloning minority-labeled samples. However, some minority samples chosen by ML-ROS may contain the most common labels. REMEDIAL (Resampling Multilabel datasets by Decoupling Highly Imbalanced Labels) approach (Charte et al., 2015c) was developed to deal with label concurrence by decoupling the majority and minority labels, whose degree is measured by SCUMBLE (Charte, Rivera, del Jesus, et al., 2014). This may cause additional complexity in a learning job when there are numerous pairs of examples with the same attributes but different labels.

To alleviate the issue of random oversampling, SMOTE (Chawla et al., 2002) was used in a heuristic oversampling technique. Nonetheless, class groups were frequently not well defined, and some samples from the majority class may have infiltrated the minority class space or vice versa. The first heuristic multilabel undersampling approach, MLeNN (Charte, Rivera, Jesús, et al., 2014) was released. It outperformed the random undersampling done by LP-RUS but only handled majority labels and comparable labels of its neighbors heuristically (Liu & Tsoumakas, 2019).

Another method was MLSMOTE (Charte et al., 2015b), which used samples with minority labels as seeds to produce new instances for multiple minority labels in most multilabel datasets. It was suggested for severely skewed multilabel datasets, but not for MLC algorithms that rely on local information. MLSOL (Liu & Tsoumakas, 2019) was also developed to examine the imbalance based on the local characteristics of minority samples rather than the entire dataset. It outperformed MLSMOTE in terms of performance and mistake correction, but it only concentrated on local label distribution. MLTL (Pereira et al., 2020) was adapted from the standard

Tomek Link method to manage the imbalance in the undersampling technique. It can define a Hamming distance threshold to eliminate the majority label, however, it is difficult to apply to highly concurrent imbalanced labels.

The REMEDIAL-HwR (REMEDIAL Hybridization with Resampling) approach (Charte et al., 2019) was divided into three hybrid methods: REMEDIAL-HwR-ROS, REMEDIAL-HwR-HUS, and REMEDIAL-HwR-SMT. In order to improve concurrent imbalanced multi-label classification (MLC), this study used multiple resampling methods such as ML-ROS, MLeNN, and MLSMOTE. These three hybrid strategies were not confined to specific situations or general solutions. Recently, (Liu & Tsoumakas, 2019), another approach for local label distribution was devised by embedding MLSOL and MLUL (Liu et al., 2022).

Despite advances in resampling techniques for imbalanced multi-label classification, there is still a literature gap regarding the optimal resampling method to improve the performance of pre-trained deep models for predicting tags, particularly in the context of Stack Overflow's multi-label classification problem with a large number of labels. There hasn't been much study on tag recommendations that combines pre-trained transformer-based models for dealing with the issue of multi-label categorization with thousands of Stack Overflow labels.

The pre-trained PTM4Tag (He et al., 2022) was developed specifically for the multi-label classification and compared to the previous Post2Vec as a baseline model, which includes BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), ALBERT (Lan et al., 2019), CodeBERT (Feng et al., 2020). There has been no prior research mentioning pre-trained ELECTRA for tag suggestion in multi-label classification challenges from Stack Overflow posts.

This work attempts to address the issue of imbalanced data by evaluating several multi-label resampling approaches to improve the performance of a deep learning pre-trained model. We concentrate primarily on multi-label classification and the efficiency of ELECTRA, the pre-trained model because ELECTRA can be trained on a single GPU due to our constrained resources. It has never been trained for tag prediction in multi-label classification problems on the Stack Overflow Q&A Site.

Chapter IV
METHODOLOGY

4.1 Pre-processing

The experiment was set up on Jupyter Notebook (anaconda3), Python 3. We choose the Stack Overflow 430,576-row dataset between the years 2009 to 2015 from the SO dump[1], which includes components such as Title, Body (description), and Code snippets, as well as thousands of tag labels. The code snippets are separated from the body, and the title and body are blended into the text after eliminating HTML tags, punctuation, and stopwords from the postings. Except for the code samples, all text is transformed to lowercase.

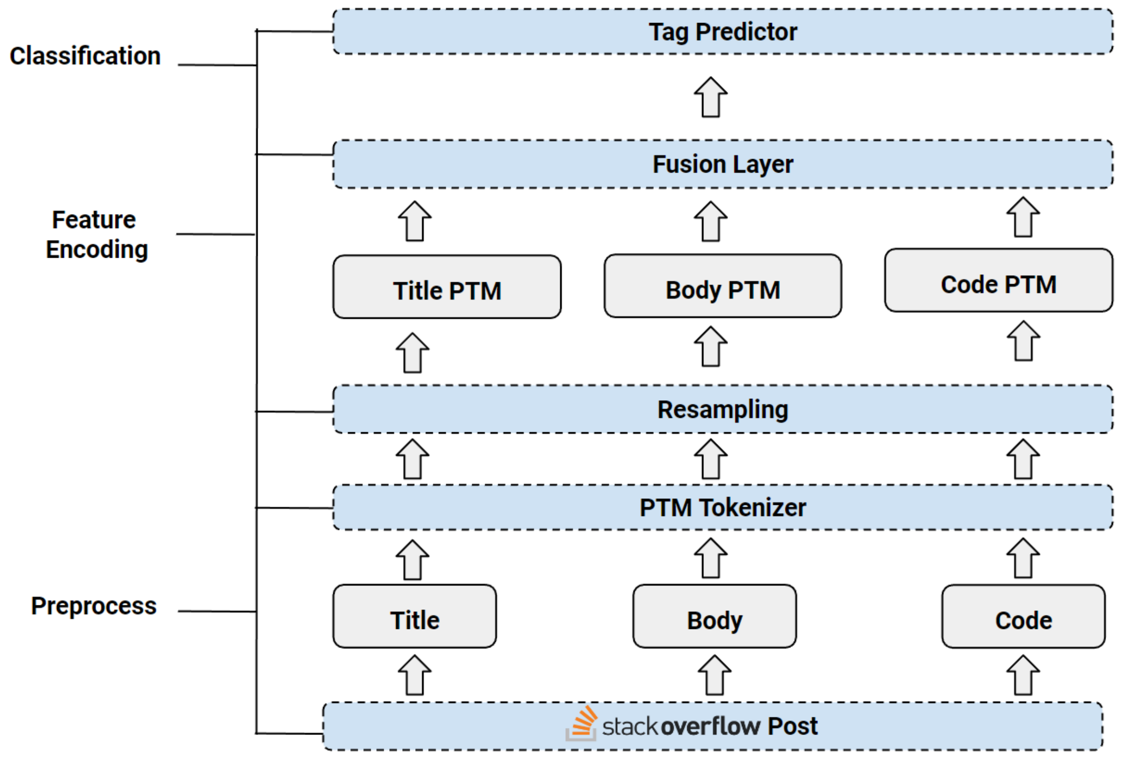


Figure 2: A workflow of Stack Overflow's tag classification

To decrease noisy data, we select just the questions covered by the most frequent tags and remove rare tags that occur less than 50 times. This process is

[1] https://archive.org/download/stackexchange

similar to the previous approach (Xu et al., 2022), which deleted extraneous attributes such as rare tags that are unimportant in providing representative tags to users for big software information sites datasets. Rare and frequent labels in a multi-label dataset limit the efficacy of resampling procedures (Feng et al., 2020).

4.2 PTM-Oriented Tokenization

The dataset was randomized and classified into three groups: 90% for a train set, 5% for a validation set, and 5% for a test set. Following that, the stratified approach was used. To extract tokens from the sentences, we utilized the same pre-trained tokenizer (see Figure 2). By transforming the texts into tokens, we explored ELECTRA. Tokenization tokens were merged with unique tokens: *<CLS> (Classification)*, the first token in each input sequence, and *<SEP>* (*Separator*), the last token in each input sequence.

4.3 Resampling Techniques

After the preprocessing and tokenization processes, we examined the pre-trained model, ELECTRA. To address the issue of imbalanced data, we used numerous multi-label resampling strategies that were suitable for our pre-trained models, including ML-ROS, MLSMOTE, MLeNN, MLTL, MLSOL, and REMEDIAL. To improve the efficiency of processing imbalanced data, several resampling strategies were applied before the training phase. LP-ROS is removed from our work since its algorithm's pseudocode is insufficient to avoid coding misunderstandings. We analyzed and compared the performance of the approaches used after balancing the data. Define L as the total set of labels, L_l for the l-th label in this set, and Y_i for the labelset associated with the i-th sample in multi-label dataset D. The same labelset can appear in several samples of D. The threshold is used to cut the set of labels into minority and majority labels. The labels whose $IRLbl$ is larger than $MeanIR$ are called majority labels. On the other hand, the labels whose $IRLbl$ is less than $MeanIR$ are called minority labels (Charte et al., 2015a). But some approaches consider for each label, a label l is defined as a minority when $IRLbl(l)$ is above $MeanIR$ (Charte et al., 2015b).

4.3.1 ML-ROS

ML-ROS (Charte et al., 2015a) (Multi-Label Random Oversampling), ML-ROS differs from LP-based approaches in that it uses individual labels to identify minority observations rather than label-sets to identify minority observations. Minority labels are identified as *IRLbl* > *MeanIR* labels. The number of samples generated is determined by a *P*% (user-defined) increase in the overall size of the dataset. The first approach is to find every minority label, that is, every label with *IRLbl* > *MeanIR*. Minority bags are used to hold observations with labels. Every observation with a minority label has a bag that contains all the observations with that label. The next stage is to oversample these minority bags. The *P*% increase in dataset size defines how many samples to clone or oversample is calculated by the *P*% increase in dataset total size. When an observation is oversampled, the number of samples to clone decreases by one when the number of samples to clone is more than zero. We loop through the label bags, oversampling one random observation from each bag. When the *IRLbl* of a label no longer exceeds the *MeanIR*, since it is no longer a minority label, a bag is removed from the bags to oversample. Sample to clone is estimated by the *P*% increase in the overall size of the dataset, as illustrated in Algorithm 1 and Figure 3.

Algorithm 1. ML-ROS algorithm pseudo-code.

```
Inputs: D: Dataset, P: Percentage
Outputs: Preprocessed dataset
1: samplesToClone = len(D) /100 *P        ▷ P% size increment
2: L = lelabelsInDataset(D) * Obtain the full set of labels
3: MeanIR = calculateMeanIR(D, L)
4: for each label in L do                  ▷ Bags of minority labels samples
5:     $IRLbl_{label}$ = calculatelRperLabel(D, label)
6:     if $IRLbl_{label}$ > MeanIR then
7:         $minBag_{i++} = Bag_{label}$
8:     end if
9: end for
10: while samplesToClone > 0 do            ▷ Instances cloning loop
11:    ▷ Clone a random sample from each minority bag
12:    for each $minBag_i$ in minBag do
13:        x = random(1, len($minBag_i$))
14:        cloneSample($minBag_i$, x)
15:        if $IRLbl_{minBag}$ <= MeanIR then
16:            $minBag = minBag_i$              ▷ Exclude from cloning
```

```
17:             end if
18:     samplesToClone
19:     end for
20: end while
```

The ML-ROS algorithm's pseudocode from (Charte et al., 2015a)

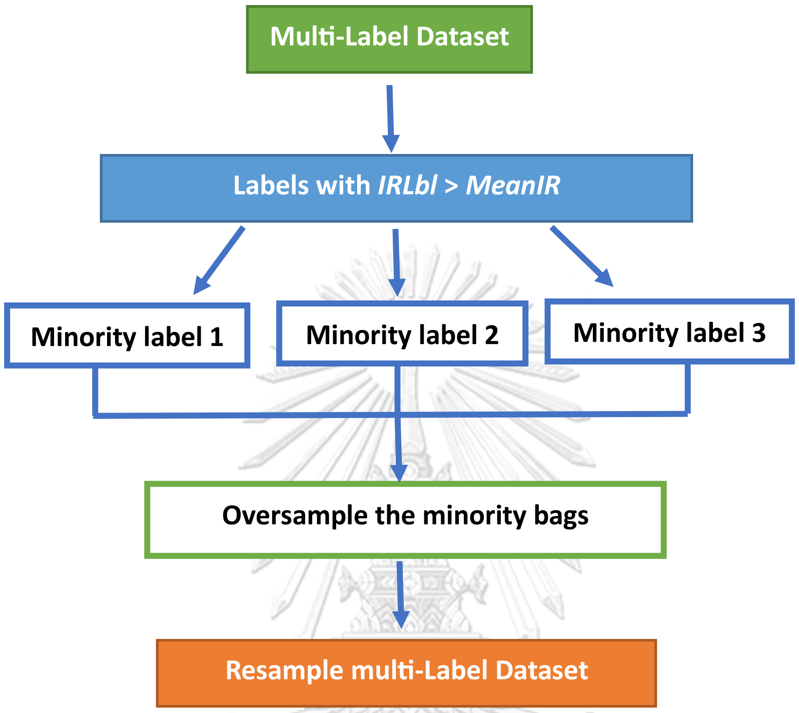

Multi-Label Dataset
Labels with *IRLbl* > *MeanIR*
Minority label 1
Minority label 2
Minority label 3
Oversample the minority bags
Resample multi-Label Dataset

Figure 3: A workflow of ML-ROS modified from Charte et al., 2015a & Kotze, 2022

4.3.2 MLSMOTE

In the Multi-label Synthetic Minority Oversampling Technique or MLSMOTE (Charte et al., 2015b), each label is examined to verify if it belongs to a minority group. A minority label is defined as *IRLbl* > *MeanIR*. If the label is a minority label, place all the observations that belong to it in a bag. For each minority bag observation, find the k nearest neighbors. Choose one of the k neighbors at random, adapted from SMOTE (Chawla et al., 2002) as shown in Figure 4.

Create a new synthetic observation with the minority observation and the random neighbor as parameters using. Interpolate the characteristics between the minority bag observation and the neighboring bag observation and assign these features to the synthetic observation. Generate the labelset for the new synthetic instance and assign the labelset to the new observation. Add the new synthetic observation to the dataset. Repeat for all labels. If the *IRLbl* of a label reaches the *MeanIR*, this label is no longer oversampled as shown in Algorithm 2-3 and Figure 5.

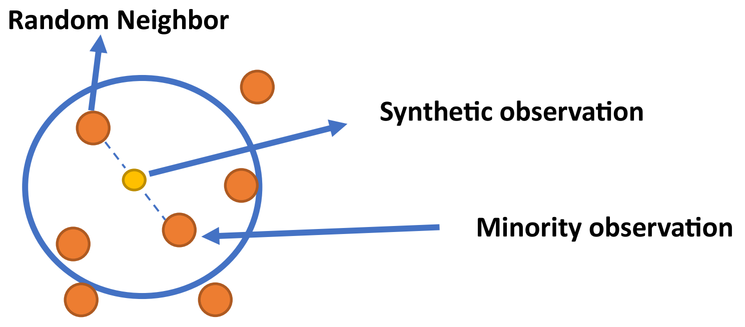


Figure 4: SMOTE from Chawla et al., 2002

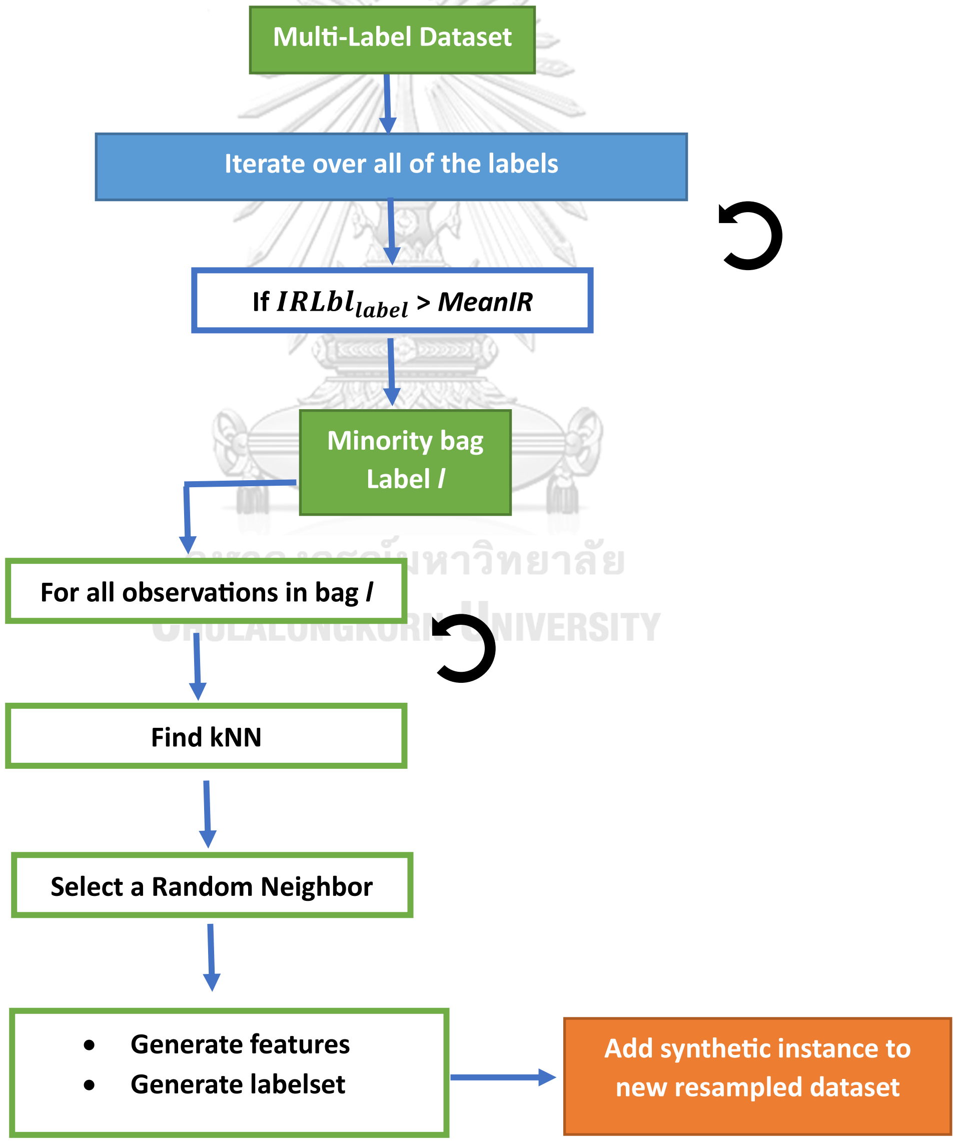

Multi-Label Dataset
Iterate over all of the labels
If $IRLbl_{label}$ > MeanIR
Minority bag Label *l*
For all observations in bag *l*
Find kNN
Select a Random Neighbor
Generate features
Generate labelset
Add synthetic instance to new resampled dataset

Figure 5: A workflow of MLSMOTE modified from Charte et al., 2015b & Kotze, 2022

Algorithm 2. MLSMOTE algorithm pseudo-code.

```
Inputs: D: Dataset to oversample, k: Number of nearest neighbors
1: L = labelsInDataset(D)                ▷ Full set of labels
2: MeanIR = calculateMeanIR(D,L)
3: for each label in L do
4:     $IRLbl_{label}$ = calculatelRperLabel(D, label)
5:     if $IRLbl_{label}$ > MeanIR then
6:         ▷ Bags of minority labels samples
7:         minBag = getAllInstancesOfLabel(label)
8:         for each sample in minBag do
9:             distances = calcDistance(sample, minBag)
10:            sortSmallerToLargest(distances)
11:            ▷ Neighbor set selection
12:            neighbors = getHeadItems(distances, k)
13:            refNeigh = getRandNeighbor(neighbors)
14:            ▷ Feature set and labelset generation
15:            synthSmpl = newSample(sample, refNeigh, neighbors)
16:            DD + synthSmpl
17:        end for
18:    end if
19: end for
```

Algorithm 3. MLSMOTE algorithm pseudo-code.

```
20: function NEWSAMPLE(Sample, refNeigh, neighbors)
21:    synthSmpl = new Sample                ▷ New empty instance
22:    ▷ Feature set assignment
23:    for each feat in synthSmpl do
24:        if typeOf(feat) is numeric then
25:            diff = refNeigh.feat - sample.feat
26:            offset = diff * randInInterval(0,1)
27:            value = sample.feat + offset
28:        else
29:            value = mostFreqVal(neighbors.feat)
30:        end if
31:            syntSmpl.feat = value
32:    end for
33:    ▷ Label set assignment
34:    IblCounts = counts(sample.labels)
35:    IblCounts+ = counts(neighbors.labels)
36:    labels = IblCounts > (k+1)/2
37:    synthSmpl.labels = labels
38:    return synthSmpl
39: end function
```

The MLSMOTE algorithm's pseudocode from (Charte et al., 2015b)

4.3.3 MLeNN

MLeNN (Charte, Rivera, Jesús, et al., 2014), the first heuristic multi-label undersampling technique, performed noticeably better than the random undersampling used by LP-RUS (Liu & Tsoumakas, 2019). The multi-label dataset will be iterated and selected for those whose labelset does not contain any labels with *IRLbl* > *MeanIR* as the candidates *C*. In this method, all instances bearing a minority label will be retained. Select a sample candidate *C*, all of them were subjected to MLeNN with nearest neighbors (NN) = 3 (3 neighbors) and Threshold (TH) = 0.75 (75% labelset difference threshold), If the *C* class differs from the class of at least half of its neighbors (that is, 2 when NN = 3), *C* should be removed as defined in Algorithm 4 and Figure 6.

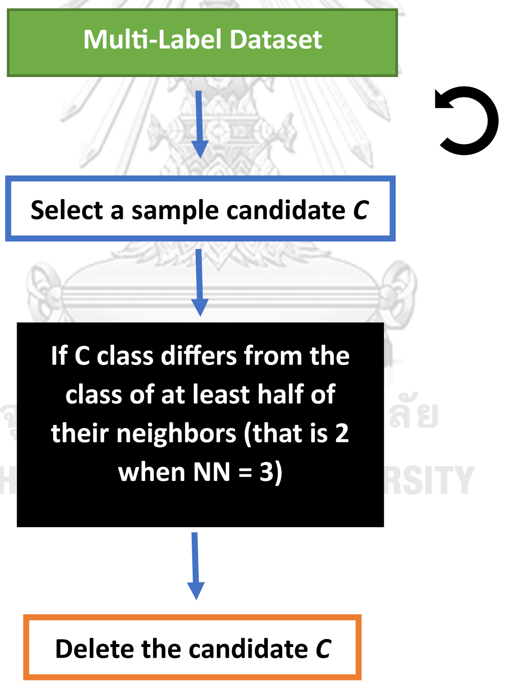

Multi-Label Dataset
Select a sample candidate C
If C class differs from the class of at least half of their neighbors (that is 2 when NN = 3)
Delete the candidate C

Figure 6: A workflow of MLeNN modified from Charte, Rivera, Jesús, et al., 2014

Algorithm 4. MLeNN algorithm pseudo-code.

```
Inputs: $D$: Dataset to resample, $TH$: Threshold, $NN$: NumNeighbors
Outputs: Preprocessed dataset
1: for each sample in $D$ do
2:     for each label in getLabelset($D$) do
```

```
3:            if IRLbl($label$) > $MeanIR$ then
4:                Jump to next sample   ▷ Preserve instance with minority labels
5:            end if
6:        end for
7:        $numDifferences = 0$
8:        for each neighbor in nearestNeighbors (sample, $NN$) do
9:            if adjustedHammingDist (sample, neighbor) > $TH$ then
10:       $numDifferences = numDifferences+1$
11:           end if
12:       end for
13:       if $numDifferences > NN/2$ then
14:           markForRemoving(sample)
15:       end if
16: end for
17: deleteAllMarkedSamples($D$)
```

The MLeNN algorithm's pseudocode from (Charte, Rivera, Jesús, et al., 2014)

4.3.4 MLTL

MLTL (Pereira et al., 2020) was modified from the traditional Tomek Link algorithm, which is finding a pair of two samples from different classes to control the undersampling technique's imbalance or cleaning method, which can specify a threshold for Hamming distance to eliminate the majority label. Figure 7 shows the type of selection for the undersampling or cleaning method in the resampling dataset as defined in Algorithm 5. A pair of two different samples from dissimilar classes are defined as the Tomek Link pair.

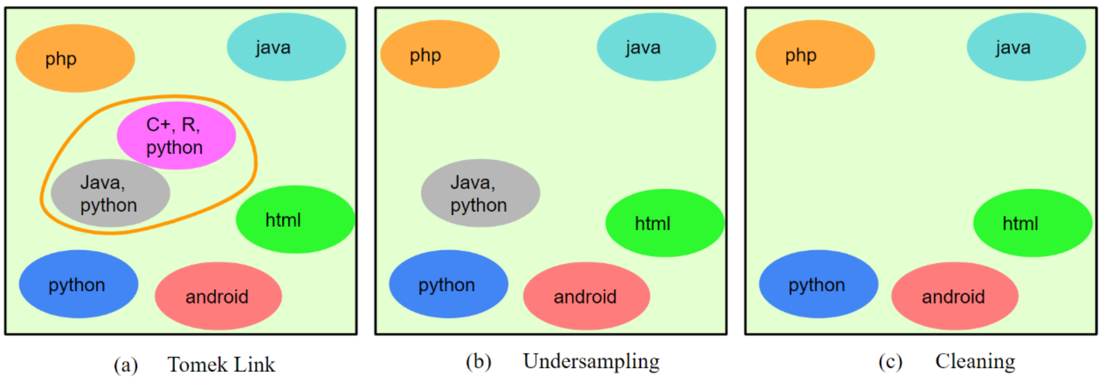


Figure 7: The Multi-Label Tomek Link was modified from Pereira et al., 2020

Labels are divided into majority bags for the undersampling approach. As a result, all observations containing a majority class label are placed in the majority bag of the relevant label. The updated Hamming Distance between each observation in the majority bags and its nearest neighbor is determined. If the adjusted Hamming Distance is greater than Threshold, the observation is added to the Tomek-Links array. Only the majority class observation is added to the set, not the nearest neighbor. An observation is not verified twice. As a result, if an observation belongs to more than one majority label, it is not rechecked. All of the majority of class observations in the Tomek-Links set is removed from the dataset as defined in Algorithm 6 and Figure 8.

A similar strategy is used for cleaning. However, rather than only checking observations from the majority classes, all observations are checked. As a result, we calculate the updated Hamming Distance from each observation to its nearest neighbor and see if it is more than the threshold. If the updated Hamming Distance is greater than the threshold, the observation and its nearest neighbor are added to an array of Tomek-Links marked for removal, as defined in Algorithm 7 and Figure 8.

Algorithm 5. Multi-Label Tomek Link.

```
Inputs: D: Dataset to resample, TH: Threshold
Output: D': Resampled dataset
1: TL = new empty list of instances
2: if (MLTL was chosen as a cleaning procedure) then
3:      TL = CLEANINGMETHOD (D, TH)
4: else
5:      TL = UNDERSAMPLINGMETHOD (D, TH)
6: end if
7: D' = new empty dataset
8: for each sample in D do
9:      if (sample not in TL) then
10:             D' = D' U sample
11:     end if
12: end for
13: return D'
```

Algorithm 6. Undersampling Method.

```
Inputs: D: Dataset to resample, TH: Threshold
Output: TL: Tomek Link instances
1: L = LABELSINDATASET(D)
2: MeanIR  = GETMEANIR (D)
3: for each l in L do
4:      iRLBI  = GETIRLBL(l)
5:      if (iRLBI < meanIR) then
6:      majBags[l] = GETINSTANCES(I)
7:      end if
8: end for
9: TL = empty list of instances
10: checkedSamples = new empty list of instances
11: for each majBag in majBags do
12:      for each sample in majBag do
13:           if (sample in checkedSamples) then
14:                continue
15:           end if
16:           NN = NEARESTNEIGHBOR(Sample)
17:           checkedSamples = checkedSamples U sample
18:           dist - ADJUSTEDHAMMINGDIST(sample, NN)
19:           if (dist TH) then
20:                TL = TL U sample
21:           end if
22:      end for
23: end for
24: return TL
```

Algorithm 7. Cleaning Method.

```
Inputs: D: Dataset to resample, TH: Threshold
Output: TL: Tomek Link instances
1: TL = new empty list of instances
2: checkedSamples = new empty list of instances
3: for each sample in D do
4:      if (sample in checkedSamples) then
5:           continue
6:      end if
7:      NN = NEARESTNEIGHBOR(Sample)
8:      checkedSamples =  checkedSamples U sample
9:      dist =  ADJUSTEDHAMMINGDIST(sample, NN)
10:     if (dist ≥ TH) then
11:          TL = TL U sample
12:     end if
13: end for
14: return TL
```

The MLTL algorithm's pseudocode from Pereira et al., 2020

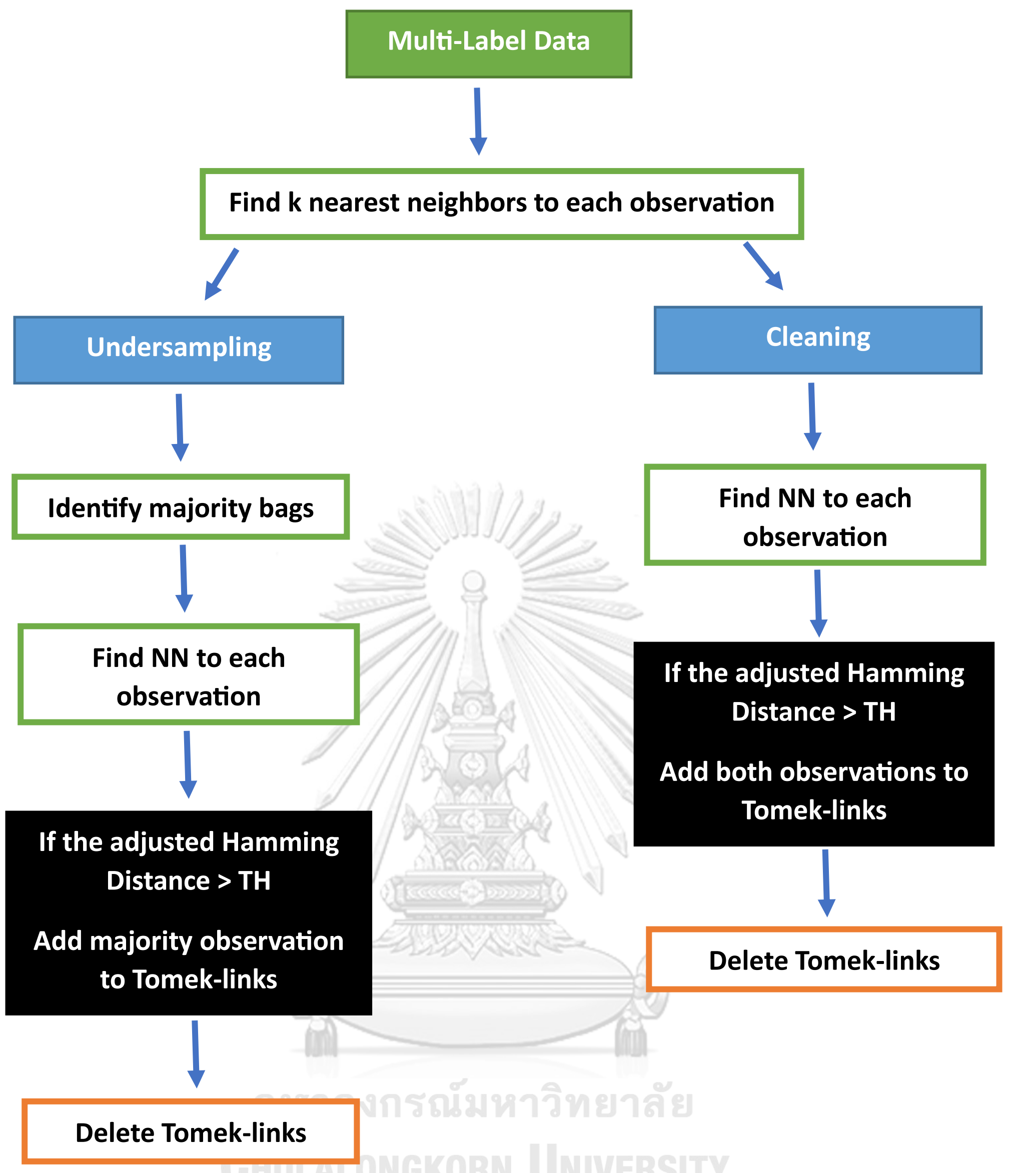

Multi-Label Data
Find k nearest neighbors to each observation
Undersampling
Cleaning
Identify majority bags
Find NN to each observation
If the adjusted Hamming Distance > TH
Add majority observation to Tomek-links
Delete Tomek-links
Find NN to each observation
If the adjusted Hamming Distance > TH
Add both observations to Tomek-links
Delete Tomek-links

Figure 8: A workflow of MLTL was modified from Pereira et al., 2020 & Kotze, 2022

4.3.5 MLSOL

MLSOL (Liu & Tsoumakas, 2019) was developed to examine the imbalance based on the regional features of minority samples as opposed to the entire dataset. While concentrating primarily on local label distribution, it outperformed MLSMOTE in terms of performance and mistake correction.

To begin, identify kNN for each observation using the Euclidean distance; the k nearest neighbors for each observation in the dataset are calculated. Then, calculating *C* is an intermediary step in establishing a sampling weight for each observation depending on its immediate surroundings.

Let *X* = $\mathbb{R}^d$ represent a *d*-dimensional input feature space, *L*= {*l1, l2, ..., lq*} represent a label set containing *q* labels, and *Y* = $\{0,1\}^q$ represent a *q*-dimensional label space. *D* = {(x_i, y_i)| 1 ≤ *i* ≤ *n*} is an n-instance multi-label training data set. Each instance (x_i, y_i) is composed of a feature vector $x_i \in X$ and a label vector $y_i \in Y$, where y_{ij} is the *j-th* element of y_i and y_{ij} = 1(0) indicates that l_j is (or is not) connected with the *i-th* instance. A multi-label technique learns from *D* the mapping function *h*: *X* $\rightarrow \{0,1\}^q$and (or) *f*: *X* $\rightarrow \mathbb{R}^d$ that, given an unknown instance *x*, outputs a label vector $\hat{y}$ containing the anticipated labels of and (or) a real-valued vector $\hat{f}\mathbf{y}$ containing the relevance degrees to x, respectively.

The *k* nearest neighbors to every observation in the dataset are calculated. The result is a *n* by *k* matrix with the distance to the *k* closest observations for each observation *i* ∈ {1,2, ..., *n*} in the dataset. Another *n* by *k* matrix is created with the index of the nearest neighbors in the original dataset as shown in equation (5).

$$C_{ij} = \frac{1}{k} \sum_{Xm \in kNN(x_i)} [[y_{mj} \neq y_{ij}]] \text{ where } C_{ij} \in \{0,1\} \quad (5)$$

The MLSOL algorithm replaces seed observations with samples. The likelihood of selecting an observation is proportional to the number of minority class observations in its immediate vicinity. *C* values vary from 0 to 1, with values near 0 as shown in equation (5) indicating a safe (hostile) neighborhood of similarly (oppositely) labeled instances. A result of C_{ij} = 1 might also be interpreted as an indication that x_i is an outlier in in comparison to l_j.

As a result, the likelihood of selecting an observation is weighted by the fraction of majority class observations in its *k* nearest neighbors. To obtain a single sampling weight w_i for each observation. C_{ij} must be aggregated for each observation. As a result, each observation x_i will be assigned a weight w_i ∈ {1,2, ... , n}. The difficulty of correctly anticipating the minority class is represented by w_i. Larger values of w_i correspond to observations that are more likely to be selected in the random sample of observations, while smaller values of w_i correspond to observations that are less likely to be selected, calculated *w* to select the random seed observations depending on w_i, Probability Proportional to Size (PPS) sampling is utilized. A uniformly distributed random number between 0 and 1 is created and multiplied by the weighted sum ($\sum_{i=1}^{n} w_i$). The randomly chosen observation is the index corresponding to the interval within which the random number falls, as shown in equation (6).

$$w_i = \sum_{k=1}^{k} \frac{C_{ij}\,[[\,y_{ij}=1\,][[C_{ij}<1]]}{\sum_{i=1}^{n} C_{ij}\,[[\,y_{ij}=1\,][[C_{ij}<1]]} \tag{6}$$

To discover observation types, the type of observation we are working with is critical for label assignment when new synthetic observations are generated. Minority observations will be classified into four categories: safe (SF), borderline (BD), rare (RR), and outlier (OT). Safe (SF) C_{ij} < 0.3 means that a region dominated by minority examples is a safe option. Borderline (BD) 0.3 < C_{ij} < 0.7 is located along the decision line between the majority and minority classes. Rare (RR) 0.7 < C_{ij} < 1 means that the majority region is located far away from the decision boundary. Outlier (OT) $C_{ij} = 1$ it is means that surrounded by instances of the majority as shown in Algorithm 8-9 and Figure 9.

Choose a random seed observation and one of the *k* nearest neighbors to the seed observation at random as a reference observation. Generate a new observation using the seed and reference observations. According to the pseudocode in algorithm 10, add the new synthetic observation to the dataset, then repeat the procedure of producing new observations until enough samples have been generated as seen in Figure 9.

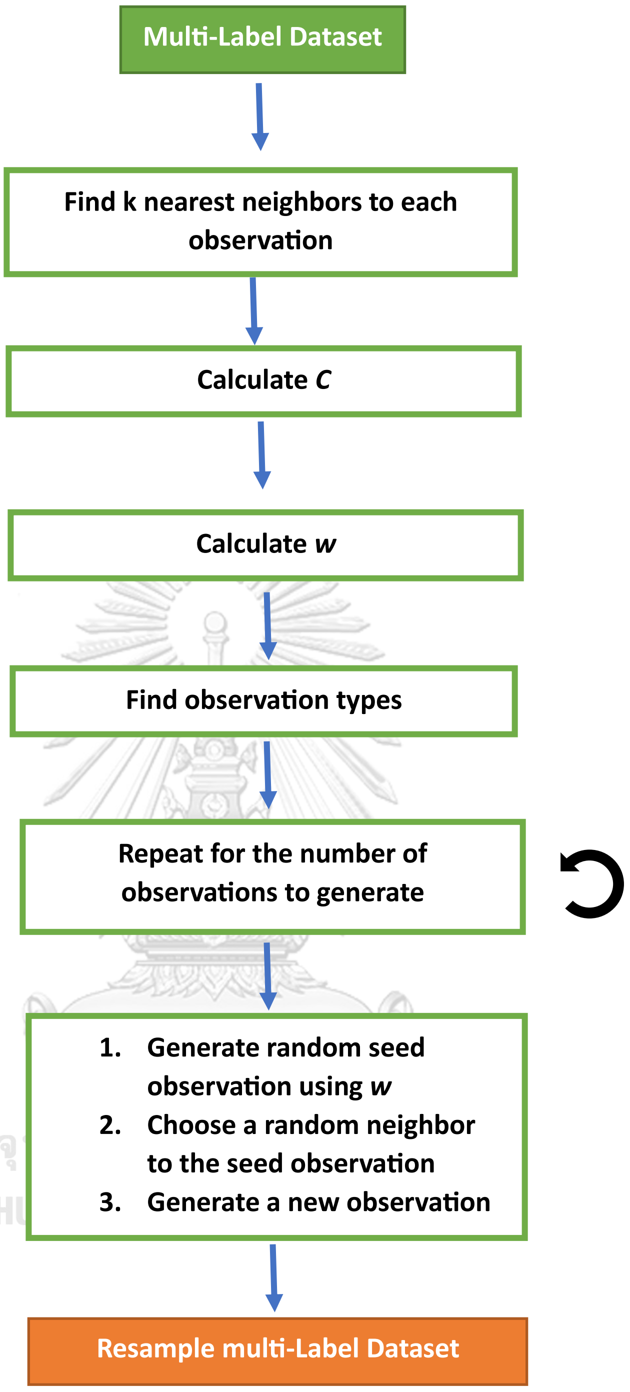

Multi-Label Dataset
Find k nearest neighbors to each observation
Calculate *C*
Calculate *w*
Find observation types
Repeat for the number of observations to generate
1. Generate random seed observation using *w*
2. Choose a random neighbor to the seed observation
3. Generate a new observation
Resample multi-Label Dataset

Figure 9: A workflow of MLSOL modified from Liu & Tsoumakas, 2019 & Kotze, 2022

Algorithm 8. MLSOL algorithm pseudo-code.

Inputs: *D*: multi-label data set, *P*: percentage of instances to be generated, *k*: Number of nearest neighbors
output: *D'*: new data set

```
1: $GenNum$ = len($D$) *$P$                ▷ Number of instances to be generated
2: $D' = D$
3: Find the kNN of each instance
```

```
4: Calculate $C$                    ▷ $C$ is the matrix storing proportion of kNNs
with opposite class for each instance and each label
5: Compute $w$
6: T =  InitTypes($C,k$)            ▷ Initialize the type of instances
7: **while** $GenNum > 0$ **do**
8:      Select a seed instance ($x_s$, $y_s$) from D based on the w
9:      Randomly choose a reference instance ($x_r$, $y_r$) from kNN
10:     ($x_c$, $y_c$) GenerateInstance (($x_s$, $y_s$), Ts, ($x_r$, $y_r$), Tr)
11:     $D' = D'$ U ($x_c$, $y_c$)
12:     $GenNum = GenNum-1$
13:     **return** $D'$
```

Algorithm 9. InitTypes.

Inputs: C: The matrix storing proportion of KNNs with opposite class for each instance and each label, k: Number of nearest neighbors
output: T: types of instances

```
1: for $i$ =1 to $n$ do                      ▷ n is the number of instances
2:     for $j$ = 1 to $q$ do                 ▷ q is the number of labels
3:         if $y_{ij}$ = majority class then
4:             $T_{ij} = MJ$
5:         else $y_{ij}$ is the minority class
6:             if $C_{ij} < 0.3$ then $T_{ij} = SF$
7:             else if $C_{ij} < 0.7$ then $T_{ij} = BD$
8:             else if $C_{ij} < 1$ then $T_{ij} = RR$
9:             else if $T_{ij} = OT$
10: repeat  re-examine $RR$ type
11:     for $i$ in 1 to $n$ do
12:         for $j$ in 1 to $q$ do
13:             if $T_{ij} = RR$ then
14:                 for each am in kNN ($x_i$) do
15:                     if $T_{ij}$= SF or $T_{ij} = BD$ then
16:                         $T_{ij}+ BD$
17:                         break
18: until no change in $T$
19: return $T$
```

Algorithm 10. GenerateInstance.

Inputs: (x_s, y_s): Seed instance, T_s: types of seed instance, (x_r, y_r): Reference instance, T_r: types of reference instance
output: (x_c, y_c): Synthetic instance

```
1: for $j$ in1 to $d$ do
2:     $x_{cj} = x_{sj}$ + Random(0, 1)*( $x_{rj} - x_{sj}$)   ▷ Random (0,1) generate a random value
       between 0 and 1
3: $d_s = (x_c, x_s)$, $d_r = (x_c, x_r)$          ▷ $dist$ return the distance between 2 instances
4: $cd = d_s/(d_s + d_r)$
```

```
5: for $j$ in 1 to $q$ do
6:     if $y_{sj} = y_{rj}$ then
7:         $y_{cj} = y_{sj}$
8:     else
9:         if $T_{sj} = MJ$ then          ▷ Ensure $y_{sj}$ being minority class
10:            $s = r$                    ▷ Swap indices of seed and reference instance
11:            $cd = 1 - cd$
12:        switch $T_{sj}$ do
13:            case $SF$ do $\theta = 0.5$ break
14:            case $BD$ do $\theta = 0.75$ break
15:            case $RR$ do $\theta = 1+1e-5$ break
16:            case $SF$ do $\theta = 0 - 1e - 5$ break
17:        if $cd \leq \theta$ then
18:            $y_{cj} = y_{sj}$
19:        else
20:            $y_{cj} = y_{rj}$
21: return $(x_t, y_t)$
```

The MLSOL algorithm's pseudocode from (Liu & Tsoumakas, 2019)

4.3.6 REMEDIAL

REMEDIAL (Resampling Multi-label datasets by Decoupling highly Imbalanced Labels) (Charte et al., 2019) method. Define *IRLbl* as the measures the imbalance level of each label instead of the total set of labels, by taking each instance D_i in a multi-label dataset D as a population, and the active labels in D_i as the individuals. If the label l is present in the instance i $\overline{IRLbl_l}$ is defined as the average imbalance level of the labels appearing in instance i.

Algorithm 11. REMEDIAL algorithm

```
1: function REMEDIAL(MLD D, Labels $L$)
2:     $IRLbli = calculateIRLbl(l \text{ in } L)$          ▷ Calculate imbalance levels
3:     $MeanIR = \overline{IRLbl_l}$
4:     $SCUMBLE_{lns_i}$ = calculateSCUMBLE($D_i$ in $D$)     ▷ Calculate $SCUMBLE$
5:     $SCUMBLE = \overline{SCUMBLE_{lns}}$
6:     for each instance $i$ in $D$ do
7:         if $SCUMBLE_{lns_i} > SCUMBLE$ then
8:             $D_i' = D_i$                                ▷ Clone the affected instance
9:             $D_i[labels_{IRLbl \leq IRMean}] = 0$       ▷ Maintain minority labels
10:            $D_i'[labels_{IRLbl > IRMean}] = 0$         ▷ Maintain majority labels
11:            $D = D + D_i'$
12:        end if
13:    end for
```

```
14: end function
```

The REMEDIAL algorithm's pseudocode from (Charte et al., 2019)

REMEDIAL shows how to deal with the concurrence of labels by separating the majority and minority labels when numerous pairs of cases have similar attributes but different labels for instance, whose level is determined by $SCUMBLE$, $SCUMBLE_{Ins} > SCUMBLE(D)$ observations will be disconnected. As a result, any observations with an above-average level of $SCUMBLE$ will be disconnected. Decoupling means that the observations will be separated into two parts. One will include all of the majority labels, while the other will include all of the minority labels. Labels with *IRLbl* ≤ *MeanIR* constitute the majority, while labels with *IRLbl* > *MeanIR* constitute the minority, which is defined in Algorithm 11 and Figure 10.

REMEDIAL is a sampling algorithm in the sense that it generates new observations. The decoupling process generates new observations while simultaneously altering previous ones. REMEDIAL distinguishes itself from other resampling algorithms by not changing the label frequencies. Even though the dataset's composition has changed, the number of observations corresponding to the majority and minority labels remains constant.

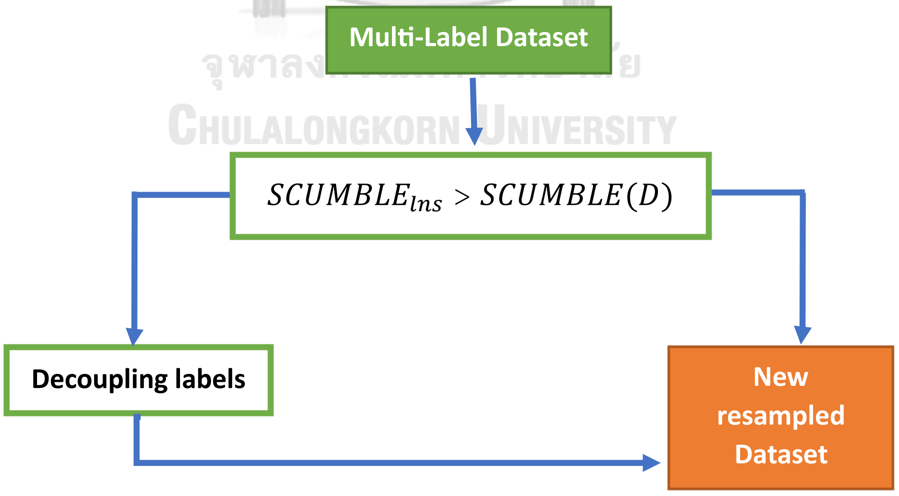


Figure 10: A workflow of REMEDIAL was modified from Charte et al., 2019 & Kotze, 2022

4.4 Model Training

The model was optimized using a constant learning rate of 7E-5 with a batch size of 64, utilizing the Adam optimizer. The objective function was set as binary-cross entropy loss. The best model was selected based on the lowest loss achieved on the validation set.

4.5 Evaluation Metrics

4.5.1 Precision@k

The average ratio of predicted ground truth tags among the list of the top-k recommended tags. Let the ground truth tags of a post as *G* for the *i-th* post in the test set and find top-k tags by Tag_i^k that can be defined as equations (7) and equations (8), respectively.

$$Precision@k_i = \frac{|GT_i \cap Tag_i^k|}{k} \quad (7)$$

For average all the values of $Precision@k_i$:

$$Precision@k = \frac{\sum_{i=1}^{|X|} Precision@k_i}{|X|} \quad (8)$$

4.5.2 Recall@k

The proportion of correctly predicted ground truth tags found in the list of ground truth tags with the same equation is followed by (Xu et al., 2022) which can be defined as equations (9) and equations (10), respectively.

$$Recall@k_i = \begin{cases} \frac{|GT_i \cap Tag_i^k|}{k} & \text{if} \quad |GT_i| > k \\ \frac{|GT_i \cap Tag_i^k|}{|GT_i|} & \text{if} \quad |GT_i| \leq k \end{cases} \quad (9)$$

For average all the values of $Recall@k_i$:

$$Recall@k = \frac{\sum_{i=1}^{|X|} Recall@k_i}{|X|} \quad (10)$$

4.5.3 F1-score@k

Define harmonic mean of $Precision@k_i$ and $Recall@k_i$ that can be defined as equations (11) and equations (12), respectively.

$$F1\text{-}score@k_i = 2 \times \frac{\text{Precision}@k_i \times \text{Recall}@k_i}{\text{Precision}@k_i + \text{Recall}@k_i} \quad (11)$$

For average all the values of $F1\text{-}score@k_i$:

$$\textit{F1-score@k} = \frac{\sum_{i=1}^{|S|} F1\text{−}score@k_i}{|X|} \qquad (12)$$

4.5.3 Area Under the Curve

To deal with the imbalance class, we use Area Under the Curve (AUC) to evaluate the model by the Area under the ROC (Receiver Operating Characteristic) curve, which consists of a True Positive Rate and False Positive Rate that can be defined as equations (12) and equations (13), respectively.

$$True\ Positive\ Rate = \frac{True\ Positive}{True\ Positive + False\ Negative} \qquad (12)$$

$$False\ Positive\ Rate = \frac{False\ Positive}{False\ Positive + True\ Negative} \qquad (13)$$

Chapter V
DISCUSSION

5.1 Generated Dataset Resampling

Before using Stack Overflow data, we set up the resampling procedures into the simulated dataset on Google Colab, containing 100,000 samples with 6 classes of labels. Due to the visualization from the bar graph is hard to indicate the labels changing compared to before and after applying the various resampling techniques. Hence, we visualize the labels changing from each class by using the scatter plot to depict the distribution of circles. The size of the circle indicates the number of labels for each class. The size of the circle will extend equally for the balance data. Each color represents the total number of the label from different classes. Class 1 represents the blue color, class 2 represents the orange color, class 3 represents the green color, class 4 represents the red color, class 5 represents the purple color, and Class 6 represents the brown color, as shown in Figure 11. Figure 11 illustrates the ideal balance data, in which the labels are represented as the circles from all the classes appear the same equal size.

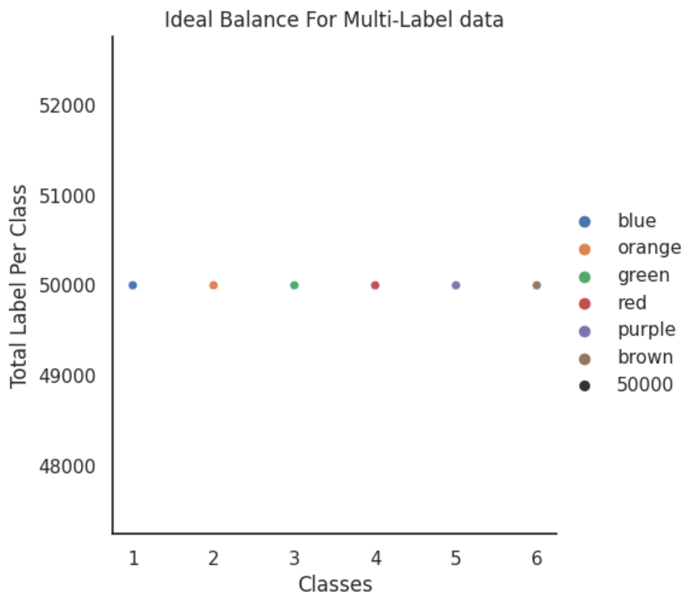
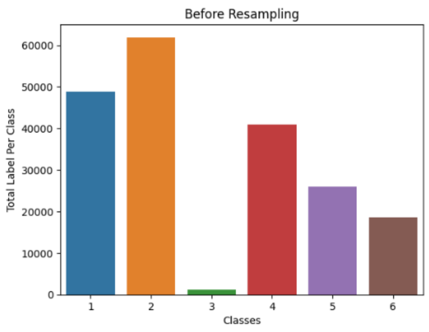
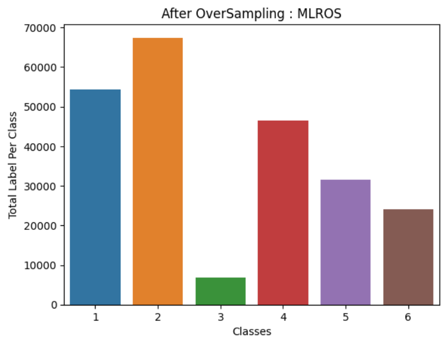


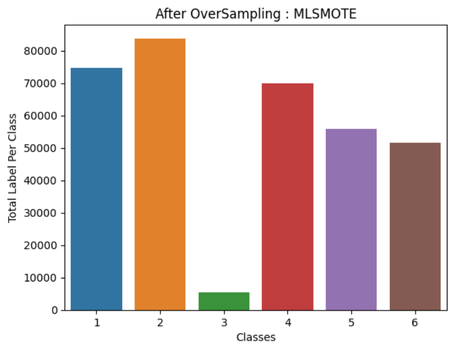
Figure 11: A scatter plot for the ideal balance of multi-label data

The generated data before applying various multi-label resampling methods are shown in Figure 12-13.

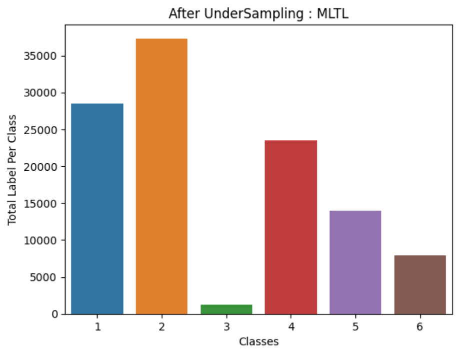


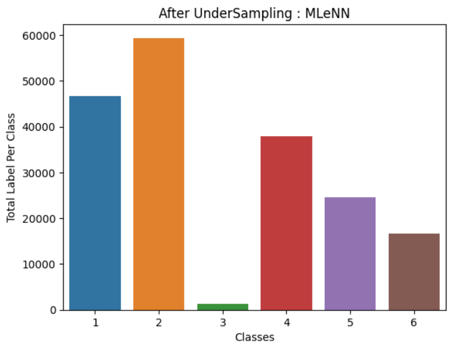
(a)



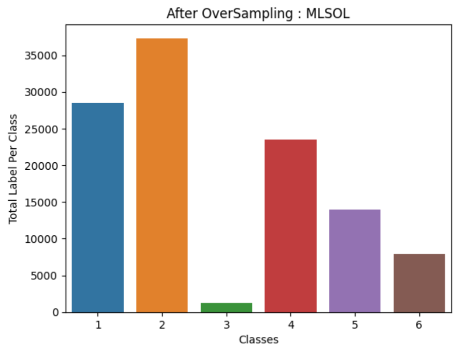


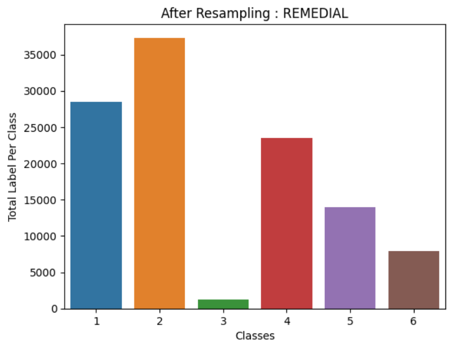
(b) (c)





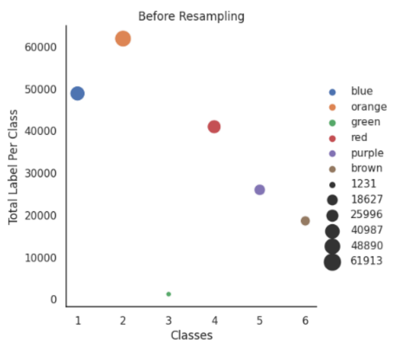
(d) (e)



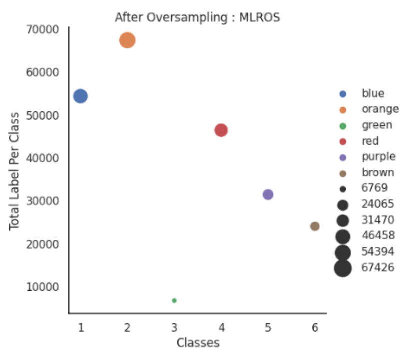


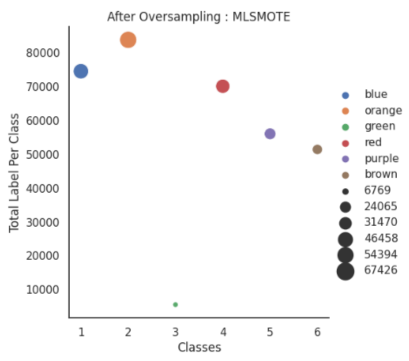
(f) (g)

Figure 12: The bar charts of the comparison between the multi-label data before and after applying various resampling methods (a) ML-ROS, (b) MLSMOTE, (c) MLTL, (d) MLeNN, (e) MLSOL, and (g) REMEDIAL

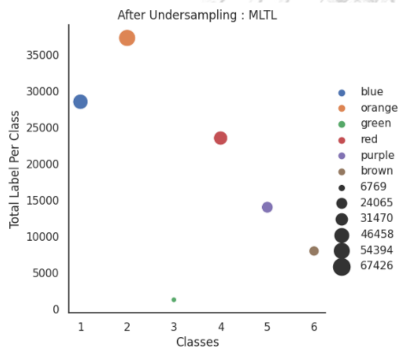


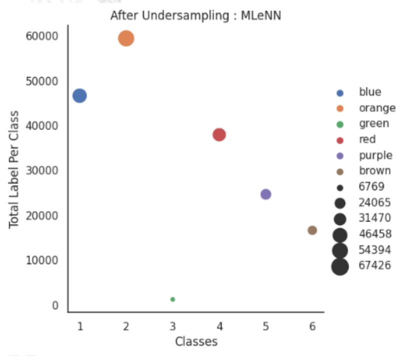
(a)



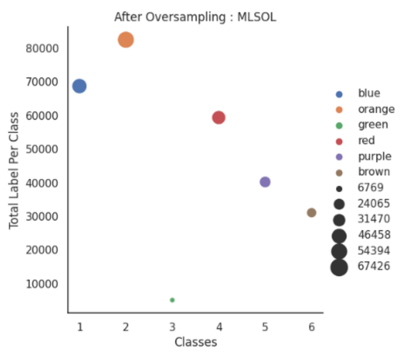


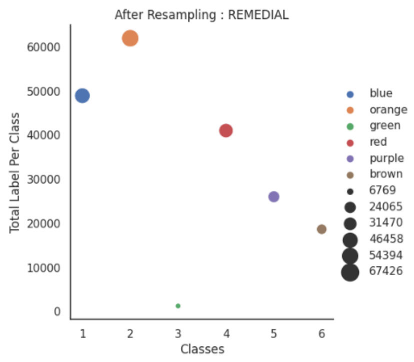
(b) (c)





(d) (e)





(f) (g)

Figure 13: The scatter plots of the comparison between the multi-label data before and after applying various resampling methods (a) ML-ROS, (b) MLSMOTE, (c) MLTL, (d) MLeNN, (e) MLSOL, and (g) REMEDIAL

Figure 12 (a) illustrates the bar graph of class imbalance before the generated data is resampled, compared to after being resampled. Each color represents the total number of labels per class. The scatter plot in Figure 13 (a) indicates the label size of the circle as the number of labels for each class, compared to the number of labels in the bar graph as shown in Figure 12 (a), the scatter plot illustrates the amount of each label from the different classes that are not the same. Before applying the various multi-label resampling methods, some classes contain over half of all labels in the dataset, but some are less than half of all labels in the dataset, the size of the circle is not the same according to the imbalances.

ML-ROS-selected minority samples may include the most prevalent labels. Majority-labeled samples are deleted, while minority-labeled samples, such as the green in class 3 are cloned, and make the green color increased, as follows by the concept of cloning the minority labels from the samples, as shown in Figure 12-13 (b). MLSMOTE generates seeds from minority label samples from all classes. The increased quantity of labels from all classes has influenced the overall label distribution, as seen by its observation, in Figure 12-13 (c). MLTL, locate a pair of two samples from distinct classes to manage the imbalance of the undersampling approach, the majority of class observations in the Tomek-Links set are all deleted from all classes, consequently, the number of labels in both graphs is reduced, as can be seen in Figure 12-13 (d). MLeNN, all occurrences with a minority label will be kept, but if the candidate's class varies from the class of at least half of their neighbors, such as class 1, they will be eliminated, which makes the number of labels in class 1 decreased, as shown in Figure 12-13 (e). MLSOL, locate the k nearest neighbors to each observation, produce a random seed observation, and pick a random neighbor to the seed observation, then generate a new observation, which increases the number of labels as seen in both graphs from Figure 12-13 (f). REMEDIAL is a sampling algorithm in the sense that it creates new observations. The decoupling process creates new observations while concurrently modifying existing ones. REMEDIAL separates itself from other resampling algorithms by not modifying the label frequencies. Even though the dataset's composition has changed, the number of observations belonging to the majority and minority labels stays consistent, As a consequence, the number of labels from all classes for REMEDIAL remains the same, as shown in Figure 12-13 (g).

Overall, the size of circles is increased after applying the multi-label oversampling resampling methods, ML-ROS, MLSMOTE, and MLSOL which represent the labels that are generated from the dataset. As a result, the number of labels is increased more than before applying the resampling methods, While the size of labels is the same for REMEDIAL, it does not work well on this generated dataset, as shown in Figure 12-13 (g), the small different change from REMEDIAL that was applied in other cases revealed the same results that REMEDIAL not good in almost every case (Charte et al., 2019). In contrast, the size of circles is reduced in MLTL and MLeNN which represent that the labels are excluded from the dataset and there are fewer amounts of each label than before applying the multi-label undersampling resampling methods.

Hence, all resampling algorithms are endorsed by the generated data. The next step is to adapt these methods to our real Stack Overflow data. However, the multi-label resampling algorithms can work differently on individual datasets (Charte et al., 2015a). For more evaluation, we use the metrics for the level of imbalance, such as IRLbl, MeanIR, and SCUMBLE, the higher *IRLbl* and *MeanIR* could increase the imbalance level, and the same as SCUMBLE, the higher value increases the inconsistent frequencies of labels.

Resampling Techniques	IRLbl	MeanIR	SCUMBLE
Before Resampling	1.26637349 1.00000000 **50.29488221** 1.51055213	9.9629	0.0431

	2.38163564 3.323831		
ML-ROS	1.23971835 1.00000000 **9.96041359** 1.45121164 2.14165026 2.80091381	3.0990	-
MLSMOTE	1.122086 1.000000 **15.467196** 1.195444 1.493664 1.632747	3.652	-
MLeNN	1.27418559 1.000000 **48.26482535** 1.56550379 2.41079326 3.57312966	9.681	-
MLTL	1.3080459 1.0000000 **30.28269699** 1.58812252 2.66423671 4.69496222	6.923	-
MLSOL	1.19925469 1.0000000 **16.02489788** 1.38878306	4.051	

	2.05047538 2.64186762		
REMEDIAL	1.26637349 1.00000000 **50.29488221** 1.51055213 2.38163564 3.323831	9.963	0.0431

Table 1: Evaluation metric for imbalance in the generated data

According to Table 1, before resampling, there is the highest number of *IRLbl* with a value of 50.295 and MeanIR with a value of 9.963. After applying various multi-label resampling methods, the level of imbalanced or MeanIR is decreased with a value of 3.099, 3.652, 9.681, 6.923, 4.051 for ML-ROS, MLSMOTE, MLTL, and MLSOL respectively. Additionally, in class 3 of labels that contained the highest imbalanced level of 50.295 before being resampled, there is a decrease of imbalance to 9.960, 15.467, 48.265, 30.283, and 16.025 for ML-ROS, MLSMOTE, MLTL, and MLSOL respectively. Apart from REMEDIAL, which is only assessed by SCUMBLE (Charte et al., 2015c), that still contains the same level of label frequency inconsistency compared to before resampling due to REMEDIAL generating a too large amount of virtual labelsets and altering the pattern of the label space.

Chapter VI
RESULTS

6.1 Experimental Results

After testing various multi-label resampling strategies for decreasing imbalanced datasets from the Q&A site, we found which multi-label resampling method is the most effective in addressing imbalanced problems in SO posts.

Model Name	Precision@K				
	P@1	*P@2*	*P@3*	*P@4*	*P@5*
ELECTRA	0.477	0.305	0.231	0.203	0.186
ML-ROS	0.503	0.320	0.237	0.209	0.187
MLeNN	0.323	0.232	0.184	0.174	0.159
MLTL	**0.517**	**0.343**	**0.251**	**0.216**	**0.193**
MLSOL	0.470	0.300	0.228	0.201	0.184
MLSMOTE	0.253	0.212	0.191	0.180	0.166
REMEDIAL	0.500	0.310	0.232	0.207	0.189

Table 2: Precision at k of resampling techniques

Model Name	Recall@K				
	R@1	*R@2*	*R@3*	*R@4*	*R@5*
ELECTRA	0.415	0.495	0.552	0.667	0.776
ML-ROS	0.429	0.515	0.563	0.683	0.779
MLeNN	0.257	0.347	0.421	0.556	0.648
MLTL	**0.444**	**0.552**	**0.602**	**0.707**	**0.804**
MLSOL	0.396	0.471	0.531	0.643	0.752
MLSMOTE	0.207	0.329	0.460	0.569	0.671
REMEDIAL	0.430	0.500	0.552	0.672	0.786

Table 3: Recall at k of resampling techniques

Model Name	F1-score@K				
	F@1	*F@2*	*F@3*	*F@4*	*F@5*
ELECTRA	0.435	0.366	0.316	0.304	0.294
ML-ROS	0.452	0.382	0.323	0.312	0.296
MLeNN	0.277	0.268	0.248	0.258	0.250
MLTL	**0.467**	**0.410**	**0.344**	**0.322**	**0.304**
MLSOL	0.419	0.354	0.308	0.298	0.289
MLSMOTE	0.221	0.248	0.262	0.265	0.259
REMEDIAL	0.452	0.371	0.317	0.318	0.298

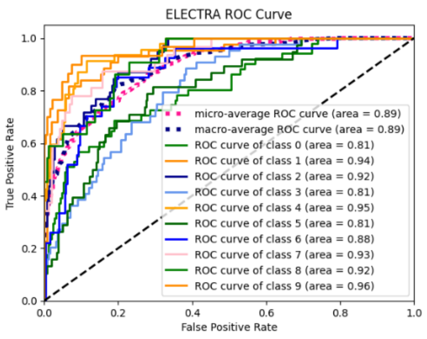
Table 4: F1-score at k of resampling techniques

At first glance, among the examples of the obtained results of all resampling techniques, MLTL obtained high-ranking values, particularly MLTL, which surpassed others in all evaluation metrics. On the other hand, MLSMOTE and MLeNN, are significantly worse than ELECTRA, ML-ROS, MLSOL, MLTL, and REMEDIAL.

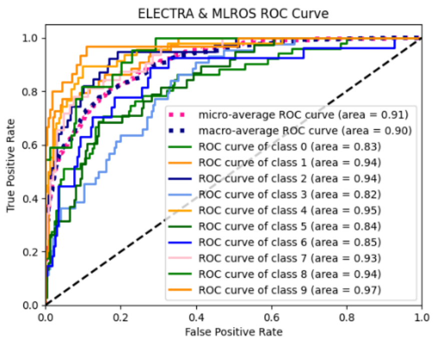
According to the observations, MLTL obtained the maximum precision value for practically every k value. MLeNN, on the other hand, had the lowest precision (see Table 2). MLTL had the strongest recall at 5 with a score of 0.804, while MLeNN had the lowest recall at 5 with a score of 0.648 (see Table 3). According to F1-score, MLSMOTE achieved the lowest performance at 1, with a score of 0.221 (see Table 4).

We applied ROC curves for additional evaluation in MLTL and MLeNN due to the imbalanced class problem. Overall, the ROC curves had a high area under the curve, indicating that MLTL had strong recall and precision. In class 3, Figure 14 (d) shows that MLTL has the largest Area under the ROC curve (0.83). Furthermore, the graph displayed a high True Positive Rate at various thresholds.

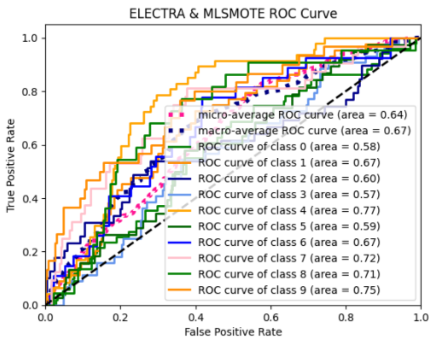
Overall, the multi-class ROC curve analysis showed that MLTL in Figure 14 (d) achieved the highest Area under the curve With values of 0.98, at class 9, while MLeNN in Figure 14 (e) performed not well in all classes, with the highest values from a score of 0.95 at class 9.



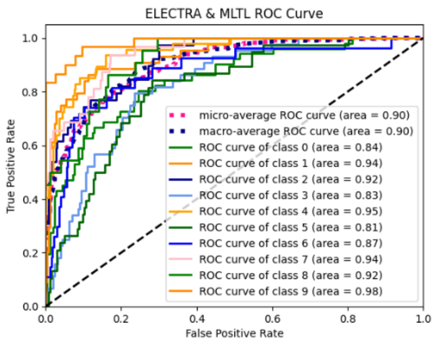
(a)

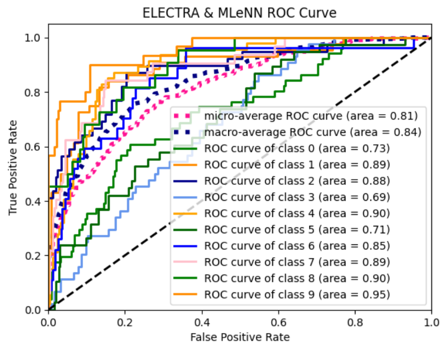


(b)

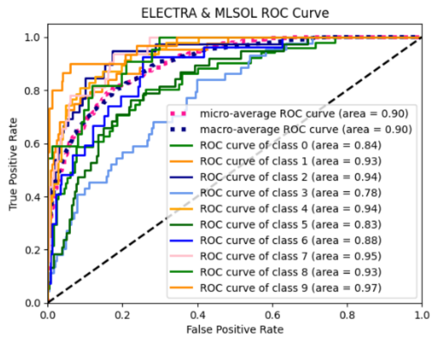


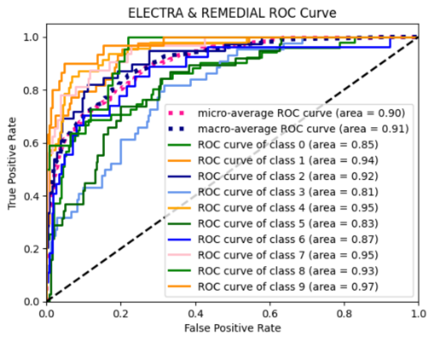
(c)





(d) (e)





(f) (g)

Figure 14: Multi-Class of the comparison between the multi-label data before and after applying various resampling methods ROC curve (a) ML-ROS, (b) MLSMOTE, (c) MLTL, (d) MLeNN, (e) MLSOL, and (g) REMEDIAL

The results of these studies imply that MLTL outperforms the other resampling techniques. The majority of labels in ML-ROS come from minority samples. REMEDIAL created an excessive number of virtual labelsets and altered the pattern of the label space. The sole focus of MLSOL is local label distribution. The class groups overlap as a result of MLSMOTE's generation of fresh synthetic samples that are added anywhere in the feature space. Because it relies on unimportant characteristics and neighbors, MLeNN performs worse than all benchmarks. It is challenging for creating an unambiguous neighbor relationship, especially for large-scale learning such as multi-label in deep learning settings. This eventually leads to the incorrect elimination of relevant samples. As a result, MLeNN has difficulty with characteristics and a neighbor-based technique that differ from those used by MLTL. To fix the imbalance, it utilizes the conventional Tomek Link technique.

6.2 Conclusion and Future Work

In order to address the imbalanced data in Stack Overflow posts, this research examines various multi-label resampling techniques and uses pre-trained ELECTRA to forecast tags. By comparing resampling methods with the pre-trained model ELECTRA on a single GPU (NVIDIA GeForce RTX 3080), this paper contributes by examining the effectiveness of various multi-label resampling techniques in improving pre-trained models to handle imbalanced data and achieving better performance metrics. According to the outcomes, MLTL is an appropriate choice to handle the imbalance in multi-label classification for our Stack Overflow data. We nevertheless advise adopting alternative approaches, such as the hybrid methods including REMEDIAL-HwR-ROS, REMEDIAL-HwR-HUS, and REMEDIAL-HwR-SMT to deal with the overfitting problem caused by the oversampling of the data. Performance can be enhanced by using additional pre-trained models, particularly when several GPUs are being used. However, researchers should select the resampling technique that is best suited for the individual dataset and specific issue.

REFERENCES

Charte, F., Rivera, A., del Jesus, M. J., & Herrera, F. (2014). Concurrence among imbalanced labels and its influence on multilabel resampling algorithms. Hybrid Artificial Intelligence Systems: 9th International Conference, HAIS 2014, Salamanca, Spain, June 11-13, 2014. Proceedings 9,

Charte, F., Rivera, A. J., del Jesus, M. J., & Herrera, F. (2019). REMEDIAL-HwR: Tackling multilabel imbalance through label decoupling and data resampling hybridization. *Neurocomputing*, 326, 110-122.

Charte, F., Rivera, A. J., Jesús, M. J. d., & Herrera, F. (2013). A First Approach to Deal with Imbalance in Multi-label Datasets. Hybrid Artificial Intelligence Systems,

Charte, F., Rivera, A. J., Jesús, M. J. d., & Herrera, F. (2014). MLeNN: A First Approach to Heuristic Multilabel Undersampling. Ideal,

Charte, F., Rivera, A. J., Jesús, M. J. d., & Herrera, F. (2015a). Addressing imbalance in multilabel classification: Measures and random resampling algorithms. *Neurocomputing*, 163, 3-16.

Charte, F., Rivera, A. J., Jesús, M. J. d., & Herrera, F. (2015b). MLSMOTE: Approaching imbalanced multilabel learning through synthetic instance generation. *Knowl. Based Syst.*, 89, 385-397.

Charte, F., Rivera, A. J., Jesús, M. J. d., & Herrera, F. (2015c). Resampling Multilabel Datasets by Decoupling Highly Imbalanced Labels. Hybrid Artificial Intelligence Systems,

Charte, F., Rivera, A. J., Jesús, M. J. d., & Herrera, F. (2017). Tackling Multilabel Imbalance through Label Decoupling and Data Resampling Hybridization. *ArXiv*, *abs/*1802.05031.

Chawla, N., Bowyer, K., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *ArXiv*, *abs/*1106.1813.

Clark, K., Luong, M.-T., Le, Q. V., & Manning, C. D. (2020). ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. *ArXiv*, *abs/*2003.10555.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep

Bidirectional Transformers for Language Understanding. *ArXiv*, *abs/*1810.04805.

Feng, Z., Guo, D., Tang, D., Duan, N., Feng, X., Gong, M., Shou, L., Qin, B., Liu, T., Jiang, D., & Zhou, M. (2020). CodeBERT: A Pre-Trained Model for Programming and Natural Languages. *ArXiv*, *abs/*2002.08155.

García, V., Mollineda, R. A., & Sánchez, J. S. (2008). On the k-NN performance in a challenging scenario of imbalance and overlapping. *Pattern Analysis and Applications*, 11, 269-280.

Giraldo-Forero, A. F., Jaramillo-Garzón, J. A., Ruiz-Muñoz, J. F., & Castellanos-Domínguez, G. (2013). Managing Imbalanced Data Sets in Multi-label Problems: A Case Study with the SMOTE Algorithm. Iberoamerican Congress on Pattern Recognition,

He, J., Xu, B., Yang, Z., Han, D., Yang, C., & Lo, D. (2022). PTM4Tag: Sharpening Tag Recommendation of Stack Overflow Posts with Pre-trained Models. 2022 *IEEE/ACM 30th International Conference on Program Comprehension (ICPC)*, 1-11.

Kotze, U. (2022). *Resampling Algorithms for Multi-Label Classification* Stellenbosch University]. Stellenbosch.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R. (2019). ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. *ArXiv*, *abs/*1909.11942.

Liu, B., Blekas, K., & Tsoumakas, G. (2022). Multi-label sampling based on local label imbalance. *Pattern Recognition*, 122, 108294.

Liu, B., & Tsoumakas, G. (2019). Synthetic Oversampling of Multi-Label Data based on Local Label Distribution. ECML/PKDD,

Liu, J., Zhou, P., Yang, Z., Liu, X., & Grundy, J. C. (2018). FastTagRec: fast tag recommendation for software information sites. *Automated Software Engineering*, 25, 675-701.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. *ArXiv*, *abs/*1907.11692.

Pant, P., Sabitha, A. S., Choudhury, T., & Dhingra, P. (2018). Multi-label Classification Trending Challenges and Approaches. *Advances in Intelligent Systems and*

Computing.

Peng, D., Gu, T., Hu, X., & Liu, C. (2021). Addressing the multi-label imbalance for neural networks: An approach based on stratified mini-batches. *Neurocomputing*, 435, 91-102.

Pereira, R. M., Costa, Y. M. G., & Silla, C. N. (2020). MLTL: A multi-label approach for the Tomek Link undersampling algorithm. *Neurocomputing*, 383, 95-105.

Tabassum, J., Maddela, M., Xu, W., & Ritter, A. (2020). Code and Named Entity Recognition in StackOverflow. Annual Meeting of the Association for Computational Linguistics,

Von der Mosel, J., Trautsch, A., & Herbold, S. (2022). On the validity of pre-trained transformers for natural language processing in the software engineering domain. *IEEE Transactions on Software Engineering*.

Wang, S., Lo, D., Vasilescu, B., & Serebrenik, A. (2014). EnTagRec++: An enhanced tag recommendation system for software information sites. *Empirical Software Engineering*, 23, 800-832.

Xu, B., Hoang, T., Sharma, A., Yang, C., Xia, X., & Lo, D. (2022). Post2Vec: Learning Distributed Representations of Stack Overflow Posts. *IEEE Transactions on Software Engineering*, 48, 3423-3441.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J. G., Salakhutdinov, R., & Le, Q. V. (2019). XLNet: Generalized Autoregressive Pretraining for Language Understanding. Neural Information Processing Systems,

Zhou, P., Liu, J., Liu, X., Yang, Z., & Grundy, J. C. (2019). Is deep learning better than traditional approaches in tag recommendation for software information sites? *Inf. Softw. Technol.*, 109, 1-13.

Zhou, P., Liu, J., Yang, Z., & Zhou, G. (2017). Scalable tag recommendation for software information sites. 2017 *IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 272-282.

VITA

NAME	Arisa Umparat
DATE OF BIRTH	24 June 1993
PLACE OF BIRTH	Bangkok, Thailand
INSTITUTIONS ATTENDED	Chulalongkorn University