International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. #, No.# (Mon-20..)

Classification of Tuberculosis Based on Chest X-Ray Images for Imbalance Data using SMOTE

Muhammad Fadhlullah [1] and Wahyono[2*]

[1] Master Program in Artificial Intelligence, Universitas Gadjah Mada, Yogyakarta, Indonesia
[2]Department of Computer Science and Electronics, Universitas Gadjah Mada, Yogyakarta, Indonesia
E-mail address: m.fadhlullah.khalilullah@mail.ugm.ac.id, wahyo@ugm.ac.id
**Corresponding author*

Received ## Mon. 20##, Revised ## Mon. 20##, Accepted ## Mon. 20##, Published ## Mon. 20##

Abstract: This research delves into the issue of dataset imbalance in the classification of Chest X-Ray (CXR) images in TBX11K by applying the Random Forest (RF) and XGBoost (XGB) methods with or without the Synthetic Minority Over-sampling Technique (SMOTE) resampling technique. The objective of this study is to assess the impact of SMOTE on model performance in the classification of CXR TBX11K images. In this research, the SMOTE technique is applied to the RF and XGB classification models. The use of SMOTE aims to increase the number of minority class samples (TB positive) to mitigate the imbalance with the majority class samples (TB negative). Each model is evaluated using the same metrics for comparison, such as accuracy, precision, recall, and F1 score. After conducting experiment, the research results indicate that the use of SMOTE technique in the RF and XGB models is effective in addressing class imbalance in the dataset. The RF model without SMOTE achieves an accuracy of approximately 93.33%, while the RF model with SMOTE achieves an accuracy of 92.72%. The XGB model without SMOTE attains an accuracy of 94.11%, whereas the XGB model with SMOTE achieves an accuracy of 94.33%. Although there is a slight decrease in accuracy in models with SMOTE during testing, the balance between precision and recall remains high. Overall, the XGB model with SMOTE is the optimal model for identifying rarely occurring positive cases, while the RF model without SMOTE is the optimal model for situations where overall accuracy is most critical.

Keywords: Tuberculosis, Random Forest, Chest X-Ray, machine learning, imbalance data, SMOTE, VGG16

1. INTRODUCTION

Tuberculosis (TB) is one of the leading causes of death worldwide [1]. According to data from the Ministry of Health of Indonesia in 2022, Indonesia ranks third after India and China in terms of the highest number of TB cases, with a total of 824,000 cases per year, out of which 93,000 result in fatalities. Early diagnosis plays a crucial role in improving the chances of recovery, preventing further spread, and significantly reducing mortality rates, in line with the WHO End TB Strategy [2] [3].

Chest X-ray (CXR) is considered the fastest and most cost-effective method among others for diagnosing TB. However, the lack of skilled radiologists in TB detection reduces its effectiveness. Additionally, the high patient burden makes the diagnostic process less accurate. Therefore, a computer-based decision support system for TB detection using CXR is needed [4] [5].

The classification of TB in CXR images using machine learning faces the problem of imbalanced data, where the number of positive class (TB) samples is much lower than the negative class (non-TB) samples. If the data is imbalanced, the trained model may tend to predict more negative cases than positive, meaning more actual TB-positive patients go undetected. This can lead to incorrect treatment and wider disease spread. The Synthetic Minority Over-Sampling Technique (SMOTE) is one method to address imbalanced data by generating synthetic samples for the minority class [6]. In this case, using SMOTE can improve the accuracy of minority class classification by reducing the false-negative rate (FN).

E-mail: wahyo@ugm.ac.id

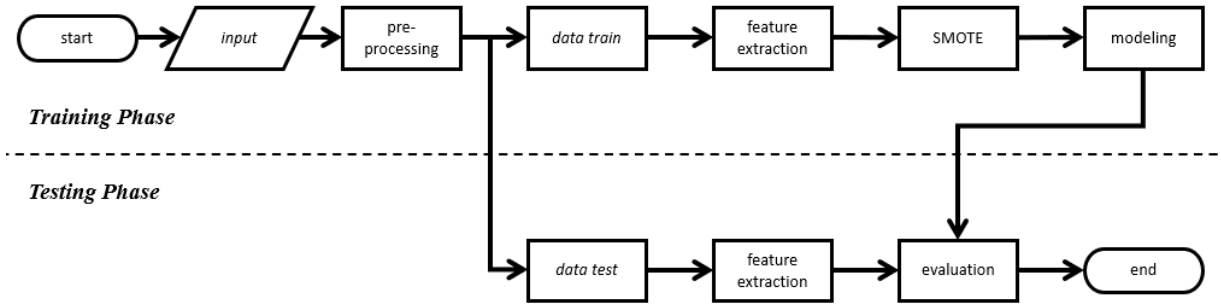

Figure 1. Research Model Algorithm.

Previous research [7] showed that SMOTE can enhance the performance of the Random Forest classification algorithm in distinguishing between the majority and minority classes. SMOTE successfully improved accuracy, sensitivity, specificity, F1-score, and AUC compared to classification without SMOTE. The success of SMOTE usage greatly depends on the ratio of the minority class presence in the data. The fewer the minority class samples, the more significant the impact of SMOTE on RF performance. However, the dataset used was numerical data to predict heart disease.

Furthermore, to extract important features from CXR images, Convolutional Neural Networks (CNNs) can be used because of their ability to automatically extract relevant features from images. CNNs have been successfully used in various image processing applications, including CXR-based TB classification. In the study by [8], TB detection was done by comparing a combination of CNN feature extraction with RF and XGBoost classification methods. The results showed that the combination of CNN for feature extraction and RF for classification achieved an accuracy of 98.667% and an AUC of 99.933% with a 90% training data percentage. However, this research was conducted on balanced CXR image data, with 6,000 data points consisting of 3,000 TB class and 3,000 normal class data points.

Therefore, this research will evaluate the effectiveness of SMOTE in addressing the data imbalance issue in TB CXR image data. CNN feature extraction and RF and XGBoost algorithms will be combined to create the classification model. Classification results will be compared between data without SMOTE resampling and data with SMOTE. The classification results to be compared include AUC and F1-score as appropriate evaluation methods for imbalanced data.

2. THE PROPOSED METHOD

A. Research Description

This study aims to assess the performance of SMOTE in addressing data imbalance in the classification of TB based on CXR images. Imbalance issues in CXR TB data can lead to classification errors and impact model evaluation results. Therefore, in this research, the SMOTE resampling method, typically used to tackle imbalance in tabular data, will be attempted to address the imbalance issue in image data. To support this, a CNN with a slightly modified VGG16 architecture will be utilized to extract features from CXR images, generating image features with dimensions suitable for processing in the SMOTE stage.

Regarding modeling, the RF and XGB classification algorithms will be employed for classification. Modeling will also be carried out without SMOTE resampling, resulting in four classification models: RF models with and without SMOTE, as well as XGBoost models with and without SMOTE. These models will be compared based on their performance using several evaluation metrics such as AUC and F1-score.

B. Data Acquisition

The study utilizes the TBX11K dataset, publicly available on Kaggle [9], containing 11,200 CXR images annotated for TB classification. It includes six classes: Healthy (5,000 images), Sick non-TB (5,000 images, lung diseases other than TB such as pneumonia or bronchitis.), Active TB (924 images, symptomatic TB), Latent TB (212 images, asymptomatic TB), Active & Latent TB (54 images, symptomatic and latent), and Uncertain TB (10 images, challenging classification). The images have a resolution of 512×512 and serve as the basis for the research.

TABLE I. CLASS AND DATA SPLIT

Category	Class	*Training*	*Testing*	Total
Non-TB	*healthy*	3000	800	3800
	sick non-TB	3000	800	3800
TB	*active TB*	473	157	630
	latent TB	104	36	140
	active & latent TB	23	7	30
TOTAL		6600	1800	8400

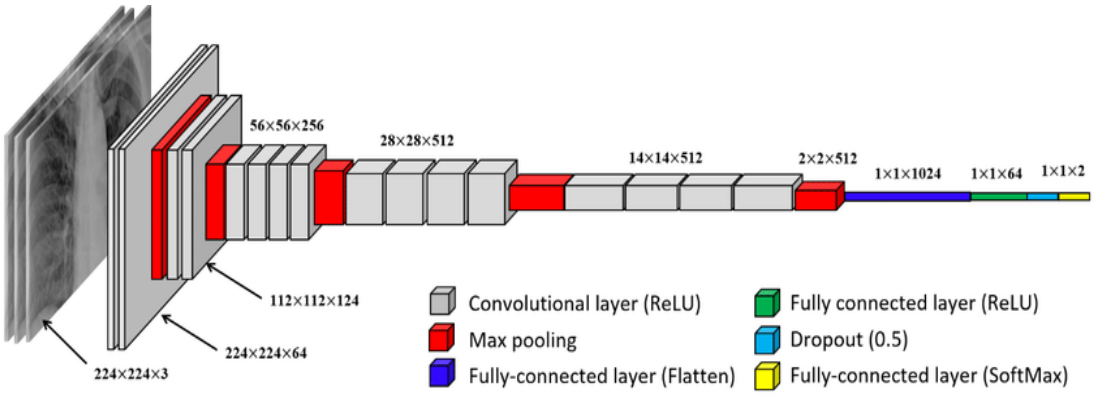

Figure 2. VGG16 Pre-trained CNN Architecture [11]

The dataset initially lacked organization, with only three folders containing image data instead of the required five classes. It also hadn't been split into training and testing data, but XML files provided filenames for both. The dataset was successfully restructured, separating training and testing data, and dividing the 'tb' folder into 'atb' for Active TB, 'ltb' for Latent TB, and 'altb' for Active&Latent TB classes. This reorganization aimed to simplify data loading. The revised dataset contained 8,400 images, as 2,790 unlabeled test images from TBX11K were excluded from this study, as show in Table I.

C. Algorithm Overview

As shown in Figure 1, the research model design begins with inputting the TBX11K CXR image dataset. The data first undergoes preprocessing to prepare it for input into the VGG16 architecture for feature extraction. Once the data is ready for processing, data splitting is performed to create a training dataset and a testing dataset. In both the training and testing phases, the data goes through the feature extraction process because the images will be represented by their features during the model creation (training phase) and prediction and evaluation processes (testing phase).

In the training phase, as this dataset experiences class imbalance, SMOTE resampling is applied to the training data to increase the number of samples from the minority class, approaching or even balancing it with the number of samples from the majority class. Subsequently, modeling is conducted using the RF and XGBoost algorithms for two different classification models. In the testing phase, the created models are evaluated using the testing dataset, and their performance is assessed using various evaluation metrics such as accuracy, precision, recall, and F1-Score.

D. Preprocessing Model

At this stage, each image data is prepared to be processed in the feature extraction stage using the VGG16 architecture. The preprocessing steps that will be performed include resizing the image data to 224×224 pixels, converting the image color space to RGB, and normalizing the pixel values of the image by subtracting the mean and dividing by the standard deviation for each color channel.

E. Feature Extraction

Feature extraction is performed using a pre-trained CNN model with the VGG16 architecture [10]. VGG16 consists of 16 layers, where the first 13 layers are convolutional and pooling layers, while the last 3 layers are fully connected layers as shown in Figure 2. The VGG16 will be slightly modified by removing the last fully connected layer, so it will only use the first 15 layers. This modification is made because the last fully connected layer is used to predict the data class, while in this research, the classification model is separate from the CNN model.

The first and second fully connected layers are responsible for preparing the image features to be used for SMOTE and classification by the classifier. This is because, in order to perform SMOTE, image features must be in the form of numerical vectors that can be measured using metrics such as Euclidean or Cosine distance. The first and second fully connected layers transform image features from a two-dimensional matrix into a one-dimensional vector by performing operations like flattening and dense connections. The flatten operation converts the matrix into a vector by arranging its elements sequentially. Meanwhile, the dense operation connects each input unit to each output unit with specific weights and adds activation functions such as ReLU or Sigmoid.

Thus, the first and second fully connected layers will produce image features that can be used for SMOTE and classification by reshaping the form and dimensions of the image features and adding weight and activation information.

F. SMOTE

The SMOTE resampling technique will be applied to the training data. Since SMOTE operates in the feature space, its output is not synthetic data that represents the actual images but rather synthetic samples that lie within the feature space. To apply SMOTE to image data, the image data has been previously prepared by VGG16 into one-dimensional vectors containing the image feature representations. SMOTE [12] will then create synthetic samples for the minority class by selecting several nearest neighbors for each sample point in the feature space and generating interpolations between these neighboring sample points. The interpolation process randomly chooses one of the neighbors and calculates the difference between the two samples. This difference is multiplied by a random number between 0 and 1 and added to the original minority sample. The result is a new sample that lies between the two minority samples. This way, SMOTE can increase the number of minority samples without eliminating variation in the image data.

There are several hyperparameters that can be modified when using SMOTE, including (1) `sampling_strategy`: Determines the ratio of the minority class to the majority class after the oversampling process. The default value is 1.0, which means the minority class will have the same number of samples as the majority class, and (2) `k_neighbors`: Specifies the number of nearest neighbors used to create synthetic samples. The default value is 5. Changing the value of `k_neighbors` aims to obtain better and more accurate synthetic results.

SMOTE hyperparameters can influence the performance of the classification model trained with oversampled data. Therefore, it's essential to optimize these hyperparameters with k-fold cross-validation to ensure that the model used for predictions is the one with the best performance.

G. Classification Model

The classification will be performed using the RF and XGB algorithms, which leverage the features generated from VGG16 extraction. RF [13] employs ensemble learning techniques by combining multiple decision trees constructed randomly to improve classification accuracy. Each tree in the ensemble is built using a random subset of training data randomly sampled from the entire dataset. During classification, the model processes the input data's features, and each tree in the ensemble generates class predictions. The final classification from the model is determined by taking the majority vote from all the trees in the ensemble. An illustration of the RF algorithm can be seen in Figure 3.

The advantages of RF include its ability to address overfitting because it divides the dataset into smaller

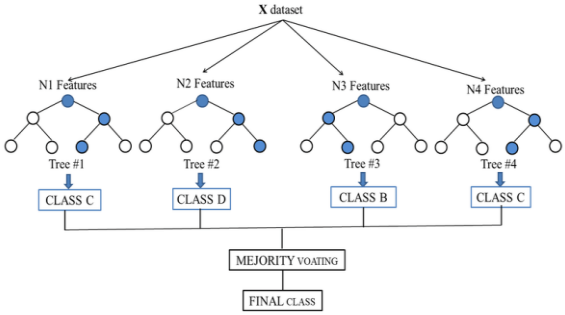

Figure 3. Illustration of Random Forest Classifier [14]

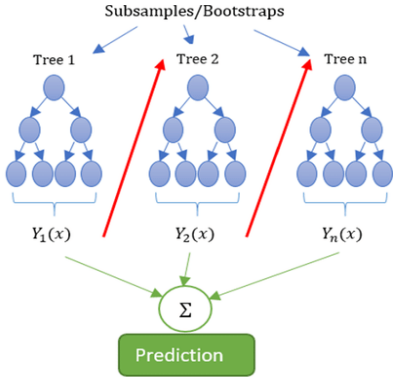

Figure 4. Illustration of XGBoost Classifier [16]

subsets and makes decisions based on these subsets. Additionally, it can handle imbalanced data problems by using different sampling techniques.

XGB [15], on the other hand, utilizes an ensemble method that combines many gradient-boosted decision trees. In image data classification, the XGB algorithm can be employed to predict the class or label of each pixel or region within an image based on the features extracted from the image. An illustration of the XGB algorithm can be seen in Figure 4.

First, XGB initializes the model with a single decision tree that partitions the image data into several groups based on image features such as color, texture, shape, and more. This decision tree is referred to as the base tree or base learner. Next, XGB calculates the loss function of the base tree by comparing the predicted class to the actual class of the image data. The loss function indicates how well the model can distinguish between different classes within the image data.

Then, XGB attempts to reduce the loss function by adding new decision trees called additional learners. These additional trees are constructed in the same manner as the base tree but use image data that has been weighted based on the loss function of the base tree. Image data with higher loss functions receive larger weights, making them more influential in the formation of the additional trees. These additional trees are then combined with the base tree using summation operations to create a new and more accurate model.

Finally, XGB repeats these steps until it reaches the predefined number of decision trees or until there is no further improvement in accuracy. The final model produced by XGBoost is a combination of numerous decision trees that complement and strengthen each other.

3. EXPERIMENT AND RESULTS

A. *Evaluation Strategy*

In building the model, the training data is divided into a 80:20 split, meaning 80% is used for training, and 20% is reserved for model validation. This is done to assess the model's performance during the hyperparameter tuning process. After validation, the next step is to perform testing using a confusion matrix. The confusion matrix is used to measure how well the model can classify data into each class. For testing purposes, classification is also done with the model without SMOTE resampling to compare the results.

Model testing is carried out by splitting the dataset into two parts: training data and testing data. Training data is used to train the model, validation data is used to determine hyperparameters and prevent overfitting, while testing data is used to objectively evaluate the model's performance. Model evaluation is done by calculating accuracy, precision, recall, and F1-score. Accuracy measures how many correct predictions were made compared to the total number of data points. Precision measures how many correct positive predictions were made compared to the total number of positive predictions. Recall measures how many correct positive predictions were made compared to the total number of actual positive data points. F1-score is the harmonic mean of precision and recall, providing a balanced measure between the two.

B. *Preprocessing and Feature Extraction Results*

Data preprocessing was carried out to prepare the data for feature extraction. The preprocessing step employed was rescaling, where each pixel value in the images was transformed into a range between 0 and 1. This was achieved by dividing each original pixel value by 255. The purpose of this was to ensure that pixel values in the images had a consistent scale, thus making the model training process more stable and convergent. Additionally, the resolution of each image data was resized to 224×224 pixels to match the input resolution requirements of the VGG16 architecture. An example of resized image data is shown in Figure 5, representing image data from the Active&Latent TB class.

Figure 5. Example of Resized Image Data

```
with tf.device('/device:GPU:0'):
#Menghasilkan output dari model untuk data gambar
    features = model.predict(image_generator)

print("Output dimensi:", features.shape)
✓ 16m 52.6s
6600/6600 [==============================] - 1009s 153ms/step
Output dimensi: (6600, 25088)
```

Figure 6. Output Dimension of Feature Extraction Results

In the feature extraction stage, both the training and testing data had their features extracted using VGG16 without the fully connected layer (FCL). This was done because the FCL layers are responsible for classifying data based on the extracted features from the preceding layers, whereas this study employed separate classification models using the RF and XGB classifier algorithms. As a result, an output dimension of (6600, 25088) was obtained, indicating that there were 6,600 image samples, with each image represented by 25,088 features. These features represent numerical representations of the images that have passed through the layers of the VGG16 model. This outcome is illustrated in Figure 6.

C. *Model Training Results*

In this stage, RF and XGB models were constructed both with and without SMOTE resampling to assess the impact of SMOTE on the performance of the RF model in predicting image data. Initially, model tuning was carried out to obtain optimal parameter configurations, and then the models were trained using these optimal configurations. The optimal configurations obtained were subsequently used as models for SMOTE tuning. The training results of models without and with SMOTE were then compared to observe the influence of SMOTE on the learning performance of the constructed models.

1) *Training Results on Random Forest*

Hyperparameter tuning was performed on the RF model with variations in the "n_estimator" parameter, controlling the number of decision trees to be built, and the "max_features" parameter, determining how many features should be considered when constructing each decision tree. In this experiment, "n_estimator (ne)" was

set to three different values: 100, 500, and 1000, while "max_features" was varied at four different percentages: 25%, 50%, 75%, and 100% of the total available features, which were 25,088 features. The results of RF model tuning were evaluated based on several evaluation metrics, including accuracy, precision, recall, and F1-Score, summarized in Table II.

TABLE II. RESULTS OF RANDOM FOREST TUNING

Parameter		Evaluation Metrics			
ne	**mf**	**accuracy**	**precision**	**recall**	**F1-score**
100	25% (6272)	93.18%	90.97%	93.18%	91.91%
100	50% (12544)	92.80%	90.66%	92.80%	91.54%
100	75% (18816)	92.73%	90.58%	92.73%	91.56%
100	100% (25088)	92.05%	89.86%	92.05%	90.81%
500	25% (6272)	93.41%	91.24%	93.41%	92.16%
500	50% (12544)	93.03%	90.89%	93.03%	91.78%
500	75% (18816)	92.35%	90.20%	92.35%	91.11%
500	100% (25088)	92.12%	89.94%	92.12%	90.91%
1000	25% (6272)	93.18%	91.00%	93.18%	91.87%
1000	50% (12544)	93.03%	90.88%	93.03%	91.76%
1000	75% (18816)	92.42%	90.27%	92.42%	91.20%
1000	100% (25088)	92.05%	89.86%	92.05%	90.83%

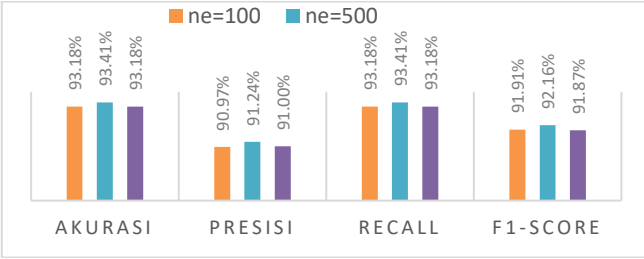

Figure 7. The impact of "n_estimator" (RF)

First, concerning the influence of the "n_estimator (ne)" parameter, as shown in Figure 7, the tuning results indicate that using 500 trees (n_estimator) leads to higher accuracy compared to 100 or 1000 trees, achieving an accuracy of approximately 93.41% when max_features=25% of the total number of features. In contrast, 100 and 1000 trees both reached an accuracy of 93.18%. This suggests that in the context of this dataset, using a larger number of trees does not always result in a significant improvement in model performance due to potential over-complexity.

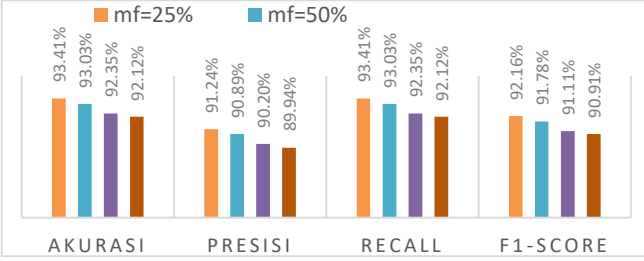

Figure 8. The impact of "max_features" (RF)

Second, concerning the influence of the "max_features (mf)" parameter, it can be observed that the higher the percentage of features used, the model's performance slightly decreases, although not significantly based on the evaluation metrics, across different "n_estimator" values. For instance, in the case of 500 trees as shown in Figure 8, using 25% of the features results in the highest performance with an accuracy of 93.41%. Subsequently, the evaluation metrics indicate a slight decrease when 50% of the features are used, resulting in an accuracy of 93.03%. Further, the metrics show a slight decrease when 75% of the features are used, yielding an accuracy of 92.35%. The lowest performance is achieved when all features (100%) are used, with an accuracy of 92.12%. These results indicate that more features do not necessarily yield better outcomes, and, on the other hand, limiting the number of features used can help prevent overfitting and improve model performance. Third, Table II shows that parameters that result in high accuracy also tend to yield good precision, recall, and F1-Score. This indicates that the model has good capabilities in classifying both positive and negative data and maintains a good balance between precision and recall.

Overall, the tuning results suggest that the Random Forest model with 500 trees and using around 25% of the features as max_features is the optimal configuration in the context of this dataset, achieving the best performance with an accuracy of approximately 93.41%. However, it's important to note that model performance does not always increase with parameter increments. There is a point where adding more estimators or maximum features no longer provides a significant improvement in performance, as observed in some cases.

TABLE III. RESULTS OF SMOTE TUNING (RANDOM FOREST)

Parameter		Evaluation Metrics			
ss	**kn**	**accuracy**	**precision**	**recall**	**F1-score**
1500	3	97.95%	97.96%	97.95%	97.95%
1500	5	98.29%	98.29%	98.29%	98.29%
1500	7	97.86%	97.86%	97.86%	97.85%
2000	3	98.46%	98.46%	98.46%	98.46%
2000	5	98.83%	98.84%	98.83%	98.83%
2000	7	98.63%	98.63%	98.63%	98.63%
2500	3	98.44%	98.45%	98.44%	98.44%
2500	5	98.33%	98.34%	98.33%	98.33%
2500	7	98.56%	98.56%	98.56%	98.55%

2) Training Results on Random Forest with SMOTE

The optimal RF configuration previously obtained, which is n_estimator=500 and max_features=25%, was applied in the SMOTE hyperparameter tuning process. The parameters optimized in this tuning stage were "sampling_strategy," controlling how many synthetic samples would be created for the minority class, and "k_neighbors," controlling the number of nearest neighbors to be considered when generating synthetic samples for the minority class. Evaluation was conducted on several combinations of "sampling_strategy" and "k_neighbors" values to assess the impact of parameter variations on model performance. The results are presented in Table III.

First, regarding the impact of the "sampling_strategy" parameter, as shown in Figure 9 with k_neighbors=3, the tuning results indicate that a "sampling_strategy" value of 2000 produces the best evaluation results with an accuracy of 98.458%, which is higher than 2500 samples with an accuracy of 98.444%. In contrast, 1500 samples resulted in the lowest performance with an accuracy of 97.952%. This suggests that increasing the number of synthetic samples generated by SMOTE can improve the model's accuracy. However, an overly aggressive "sampling_strategy" value can lead to overfitting. Therefore, from this case, it can be stated that increasing the number of minority class data should be done judiciously and does not necessarily need to equal the majority class count.

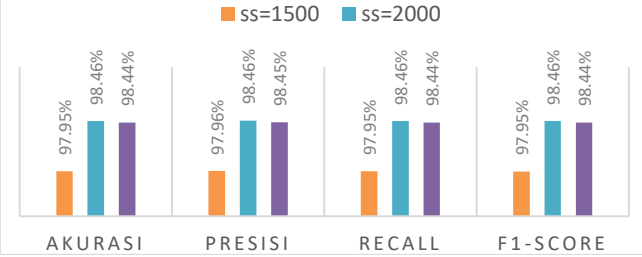

Figure 9. The impact of "sampling_strategy" (SMOTE RF)

Second, regarding the impact of the "k_neighbors" parameter, as shown in Figure 10 with "sampling_strategy"=2000, it can be observed that using 5 nearest neighbors results in the highest accuracy of around 98.833%, followed by 3 nearest neighbors at approximately 98.458%, and 7 nearest neighbors at around 98.625%. This suggests that employing a fewer or moderate number of nearest neighbors, such as 5 or 3, tends to yield better performance compared to using too many (e.g., 7). Therefore, it can be stated that a more moderate SMOTE approach can provide optimal results in boosting the minority class without causing overfitting.

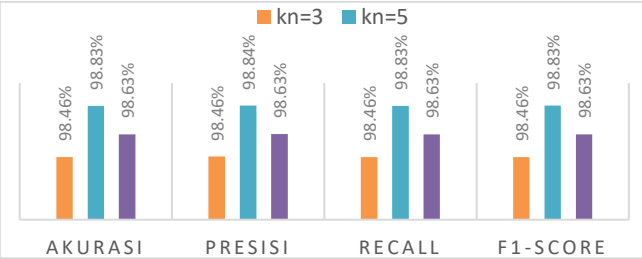

Figure 10. The impact of "k_neighbors" (SMOTE RF)

Third, based on Table III, the optimal SMOTE configuration for the RF model for this dataset is "sampling_strategy"=2000 and "k_neighbors"=5. In the overall analysis, it can be concluded that using SMOTE with the optimal "sampling_strategy" and "k_neighbors" values can significantly enhance the performance of the RF model in handling class imbalance. This results in a model that can classify data more accurately, especially for the minority class.

3) Training Results on XGBoost

Hyperparameter tuning was performed on the second model using the XGB classifier algorithm with variations in the "n_estimator" parameter, which controls the number of decision trees to be built, and the "learning_rate" parameter, which governs the influence of each tree in the ensemble on the final model's predictions. In this experiment, "n_estimator" was set to three different values: 100, 500, and 1000, and "learning_rate" was also varied with three different values: 0.01, 0.1, and 0.2. The tuning results of the XGB model were evaluated based on several evaluation metrics, including accuracy, precision, recall, and F1-Score, which are summarized in Table IV.

TABLE IV. RESULTS OF XGBOOST TUNING

Parameter		Evaluation Metrics			
ne	**lr**	**accuracy**	**precision**	**recall**	**F1-Score**
100	0.01	92.50%	90.51%	92.50%	91.40%
100	0.1	93.56%	91.36%	93.56%	92.39%
100	0.2	94.09%	91.94%	94.09%	92.95%
500	0.01	93.64%	91.47%	93.64%	92.49%
500	0.1	94.24%	92.28%	94.24%	93.22%
500	0.2	94.55%	92.45%	94.55%	93.46%
1000	0.01	93.71%	91.54%	93.71%	92.56%
1000	0.1	94.17%	92.21%	94.17%	93.16%
1000	0.2	94.55%	92.46%	94.55%	93.47%

First, regarding the impact of "n_estimator" on the XGB model, it can be observed that increasing the number of estimators generally improves the model's performance, as shown in Figure 11, which refers to one of the tuning cases with "learning_rate"=0.2. When "n_estimator" increases from 100 to 1000, there is an improvement in all evaluation metrics. Accuracy increases from 94.091% to 94.545%, precision increases from 91.938% to 92.455%, recall increases from 94.091% to 94.545%, and F1-Score increases from 92.951% to 93.466%. From these results, it can be stated that using more estimators produces a stronger XGB model in classifying data.

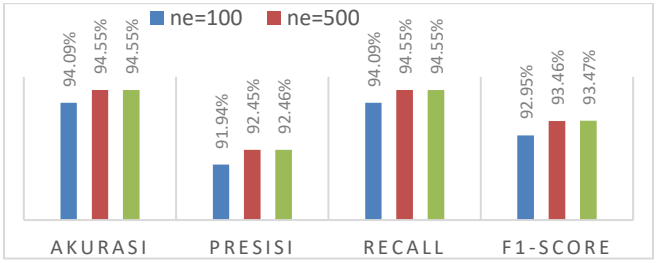

Figure 11. The impact of "n_estimator" (XGB)

Second, regarding the influence of the learning rate parameter, Figure 12 shows one of the tuning cases with "n_estimator"=1000. It can be observed that overall, the model's performance improves with increasing learning

rate values from 0.01 to 0.2. This indicates that the model tends to achieve better results when it learns faster with a higher learning_rate value. Therefore, based on these results, it can be stated that in this study, increasing the learning_rate will enhance the model's learning speed and improve its performance.

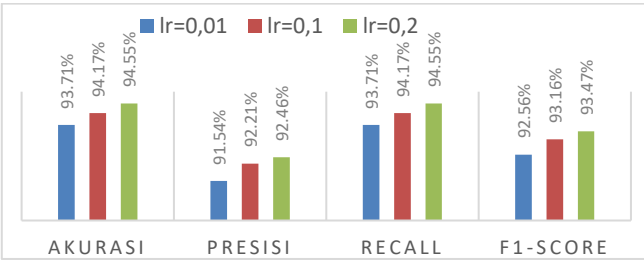

Figure 12. The impact of "learning_rate" (XGB)

Third, based on the data in Table IV, the tuning results also show good balance between precision and recall in the XGB model. Despite variations in parameter settings, the precision and recall values almost always come close to each other, reflecting that the model has the ability to correctly identify most positive instances with very few false positives. From this tuning process, the optimal configuration for the XGB model with 1000 trees and a learning rate of 0.2 is obtained.

4) Training Results on XGBOOST with SMOTE

The optimal XGB configuration obtained earlier, which is "n_estimator"=1000 and "learning_rate"=0.2, is applied in the hyperparameter tuning process with SMOTE. The parameters optimized in this tuning phase are "sampling_strategy," which controls how many synthetic samples will be created for the minority class, and "k_neighbors," which controls the number of nearest neighbors to consider when creating synthetic samples for the minority class. Evaluation is carried out for various combinations of "sampling_strategy" and "k_neighbors" values to assess the impact of these parameter variations on the model's performance. The results are shown in Table V.

TABLE V. RESULTS OF SMOTE TUNING (XGBOOST)

Parameter		Evaluation Metrics			
ss	**kn**	**accuracy**	**precision**	**recall**	**F1-score**
1500	3	98.19%	98.20%	98.19%	98.19%
1500	5	98.29%	98.30%	98.29%	98.29%
1500	7	98.43%	98.44%	98.43%	98.43%
2000	3	99.00%	99.00%	99.00%	99.00%
2000	5	99.29%	99.29%	99.29%	99.29%
2000	7	99.13%	99.13%	99.13%	99.12%
2500	3	99.15%	99.15%	99.15%	99.15%
2500	5	99.11%	99.11%	99.11%	99.11%
2500	7	99.33%	99.34%	99.33%	99.33%

First, regarding the impact of the "sampling_strategy" parameter, as shown in Figure 13 for the case when "k_neighbors"=7, the tuning results generally show improvement in each evaluation metric as the "sampling_strategy" value increases. A higher "sampling_strategy" value results in more synthetic samples being generated. This suggests that in the case of SMOTE with XGB, a more aggressive oversampling strategy can enhance the model's ability to classify imbalanced data.

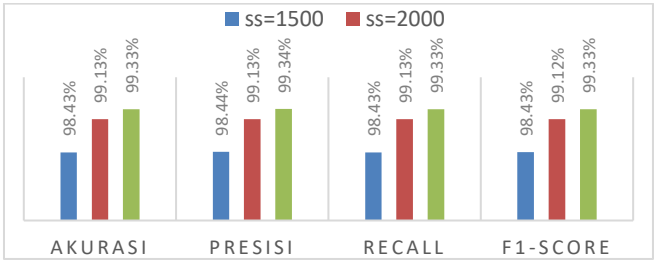

Figure 13. The impact of “sampling_strategy” (SMOTE XGB)

Figure 14. The impact of "k_neighbors" (SMOTE XGB)

Second, regarding the influence of k_neighbors, Figure 14 shows a tuning case with a sampling_strategy of 2000. The graph indicates that the best evaluation results for each evaluation metric are obtained when k_neighbors is set to 5, rather than the highest value of k_neighbors, although the differences in evaluation results are not significant. Therefore, it can be concluded that in the context of this research, the parameter k_neighbors should be set to an appropriate value, not too small or too large, to achieve optimal model performance and reduce the risk of overfitting.

Nonetheless, the tuning results indicate that all evaluation metrics have very high values and are close to each other. This indicates a good balance between the model's ability to identify positive and negative instances and the minimization of false positives and false negatives.

In the overall analysis, this experiment reveals that an XGBoost model combined with SMOTE and optimal parameter configuration can produce a highly robust and reliable classification model for addressing class imbalance issues.

5) Discussion of Training Model Results

In the training model results stage, this study built two main models, namely RF and XGB, with and without using the SMOTE resampling technique to address class imbalance in image data. The entire training model results are summarized in Table VI.

TABLE VI. MODEL TRAINING RESULTS

Model	Evaluation Metrics			
	accuracy	precision	recall	F1-Score
RF	93.33%	90.86%	93.33%	92.02%
RF+SMOTE	92.72%	90.50%	92.72%	91.59%
XGB	94.11%	91.83%	94.11%	92.95%
XGB+SMOTE	94.33%	92.24%	94.33%	93.27%

Next, in the RF model with SMOTE, the use of SMOTE with sampling_strategy=2000 and k_neighbors=5 resulted in the best performance with an accuracy of 98.83%. This indicates that SMOTE with a moderate strategy can significantly improve the model's ability to handle class imbalance without overfitting.

Furthermore, in the XGB model without SMOTE, increasing the number of estimators (n_estimator) generally improved the model's performance, with 1000 estimators and learning_rate=0.2 achieving the highest accuracy of 94.55%. Additionally, increasing the learning_rate also improved the model's performance. A good balance between precision and recall was observed in the XGB model. The optimal configuration for XGB was found to be 1000 estimators and learning_rate=0.2.

Finally, in the XGB model with SMOTE, tuning results showed that using SMOTE with sampling_strategy=2000 and k_neighbors=5 yielded the best performance with an accuracy of 99.29%. This indicates that the combination of XGB with SMOTE and optimal parameters produces a very strong and reliable model for addressing class imbalance.

Overall, both RF and XGB models can be configured with optimal parameters to address class imbalance. However, the XGB model, especially when combined with SMOTE, can achieve higher performance with very high accuracy. Therefore, in this case, the XGBoost model with SMOTE and optimized parameters is the best choice for predicting imbalanced class image data.

D. Results and Discussion on Testing Data

During the model testing phase, four different models were evaluated: RF without SMOTE, RF with SMOTE, XGB without SMOTE, and XGB with SMOTE, using the same evaluation metrics: accuracy, precision, recall, and F1 score. The summary of all model testing results is presented in Table VII.

TABLE VII. MODEL TESTING RESULTS

Model	Evaluation Metrics			
	accuracy	precision	recall	F1-Score
RF	93.33%	90.86%	93.33%	92.02%
RF+SMOTE	92.72%	90.50%	92.72%	91.59%
XGB	94.11%	91.83%	94.11%	92.95%
XGB+SMOTE	94.33%	92.24%	94.33%	93.27%

The RF model without SMOTE achieved an accuracy of approximately 93.33% when tested on unseen data, indicating good generalization capabilities. Precision and recall also had a good balance, with an F1 score of around 92.02%. This model can be considered effective in classifying data with a high level of accuracy and a good balance between precision and recall.

On the other hand, the RF model enhanced with SMOTE produced a very high accuracy, reaching 92.72% on the test data, although slightly lower than the training data. Nevertheless, the precision and recall rates remained high, with an F1 score of about 91.59%. This model remained effective in classifying the test data and provided a high level of accuracy along with a good balance between precision and recall.

As for the XGB model without SMOTE, it achieved an accuracy of around 94.11% on the test data, indicating good generalization abilities. Precision and recall rates also remained high, with an F1 score of about 92.95%. This XGB model can be considered a strong choice for classification tasks on the dataset used.

Lastly, the XGB model enhanced with SMOTE achieved an accuracy of approximately 94.33% on the test data, although slightly lower than the training data. Precision and recall rates remained high, with an F1 score of around 93.27%. This model proved to be highly effective in addressing class imbalance issues and delivered strong performance on both the training and test data.

Overall, the testing results indicate that these models can perform well in classifying data, whether with or without SMOTE, with each model having its own advantages and characteristics. Model selection depends on specific use-case goals and contexts, as well as the trade-offs between desired accuracy, precision, and recall for a particular application.

4. CONCLUSION

Based on the research findings, the issue of class imbalance in the TBX11K chest X-ray image dataset has been effectively mitigated using the SMOTE technique in both Random Forest and XGBoost models. These models exhibited good performance, accurately classifying TB-positive cases without compromising their ability to handle TB-negative data. Additionally, the Random Forest and XGBoost models without SMOTE demonstrated strong classification performance on test data. Therefore, these conclusions provide a viable solution for addressing class imbalance in this dataset, making these models suitable for confident use in lung X-ray image classification for TB detection, with a focus on positive TB cases. In terms of recommendations, future research should explore a wider range of parameter values, conduct cross-validation for model tuning, consider alternative oversampling and under sampling techniques, and potentially seek additional data to further enhance model capacity.

REFERENCES

[1] S. Urooj, S. Suchitra, L. Krishnasamy, N. Sharma, and N. Pathak, "Stochastic Learning-Based Artificial Neural Network Model for an Automatic Tuberculosis Detection System Using Chest X-Ray Images," IEEE Access, vol. 10, no. September, pp. 103632–103643, 2022, doi: 10.1109/ACCESS.2022.3208882.

[2] W. H. Organization, Global tuberculosis report 2020: executive summary. 2020. [Online]. Available: http://apps.who.int/bookorders.

[3] G. V. Sreena, N. Ponraj, and L. P. Deepa, "Study on public chest X-ray data sets for lung disease classification," 2021 3rd Int. Conf. Signal Process. Commun. ICPSC 2021, no. May, pp. 54–58, 2021, doi: 10.1109/ICSPC51351.2021.9451726.

[4] U. K. Lopes and J. F. Valiati, "Pre-trained convolutional neural networks as feature extractors for tuberculosis detection," Comput. Biol. Med., vol. 89, no. February, pp. 135–143, 2017, doi: 10.1016/j.compbiomed.2017.08.001.

[5] R. Hooda, S. Sofat, S. Kaur, A. Mittal, and F. Meriaudeau, "Deep-learning: A potential method for tuberculosis detection using chest radiography," Proc. 2017 IEEE Int. Conf. Signal Image Process. Appl. ICSIPA 2017, pp. 497–502, 2017, doi: 10.1109/ICSIPA.2017.8120663.

[6] A. Ali, S. M. Shamsuddin, and A. L. Ralescu, "Classification with class imbalance problem: A review," Int. J. Adv. Soft Comput. its Appl., vol. 7, no. 3, pp. 176–204, 2015.

[7] E. Erlin, Y. Desnelita, N. Nasution, L. Suryati, and F. Zoromi, "Dampak SMOTE terhadap Kinerja Random Forest Classifier berdasarkan Data Tidak seimbang," MATRIK J. Manajemen, Tek. Inform. dan Rekayasa Komput., vol. 21, no. 3, pp. 677–690, Jul. 2022, doi: 10.30812/matrik.v21i3.1726.

[8] A. R. Laeli, Z. Rustam, and J. Pandelaki, "Tuberculosis Detection based on Chest X-Rays using Ensemble Method with CNN Feature Extraction," in 2021 International Conference on Decision Aid Sciences and Application, DASA 2021, Institute of Electrical and Electronics Engineers Inc., 2021, pp. 682–686. doi: 10.1109/DASA53625.2021.9682237.

[9] Y. Liu, Y.-H. Wu, Y. Ban, H. Wang, and M.-M. Cheng, "Rethinking Computer-aided Tuberculosis Diagnosis." [Online]. Available: http://mmcheng.net/tb/

[10] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc., pp. 1–14, 2015.

[11] M. Y. Kamil, "A deep learning framework to detect Covid-19 disease via chest X-ray and CT scan images," Int. J. Electr. Comput. Eng., vol. 11, no. 1, pp. 844–850, 2021, doi: 10.11591/ijece.v11i1.pp844-850.

[12] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," J. Artif. Intell. Res., vol. 16, no. February, pp. 321–357, 2002, doi: 10.1613/jair.953.

[13] S. Mirsa and H. Li, "Noninvasive fracture characterization based on the classification of sonic wave travel times," Mach. Learn. Subsurf. Charact., no. 9, pp. 243–287, 2020, doi: https://doi-org.ezproxy.ugm.ac.id/10.1016/B978-0-12-817736-5.00009-0.

[14] S. Saha and S. M. M. Ahsan, "Rice Leaf Disease Recognition using Gray-Level Co-Occurrence Matrix and Statistical Features," 2021 5th Int. Conf. Electr. Inf. Commun. Technol. EICT 2021, no. December 2021, 2021, doi: 10.1109/EICT54103.2021.9733511.

[15] H. Sunata, "Komparasi Tujuh Algoritma Identifikasi Fraud ATM Pada PT. Bank Central Asia Tbk," JATISI (Jurnal Tek. Inform. dan Sist. Informasi), vol. 7, no. 3, pp. 441–450, 2020, doi: 10.35957/jatisi.v7i3.471.

[16] C. EL Mazgualdi, T. Masrour, I. El Hassani, and A. Khdoudi, "Machine learning for KPIs prediction: a case study of the overall equipment effectiveness within the automotive industry," Soft Comput., vol. 25, no. 4, pp. 2891–2909, 2021, doi: https://doi.org/10.1007/s00500-020-05348-y.

Muhammad Fadhlullah is currently active as a student in Postgraduate Degree of Universitas Gadjah Mada in Artificial Intellegence. Intermediate Python programming and graphic design skills. Possess good analytical and presentation skills. Highly motivated to learn front-end programming.

Wahyono received bachelor of computer science from Gadjah Mada University, Indonesia in 2010, and doctoral degree at the Graduate School of Electrical Engineering, University of Ulsan, Ulsan, Korea. Since 2012, He has been serving as an assistant lecturer in Department of Computer Science and Electronics, Faculty of Mathematics and Natural Science, Gadjah Mada University, Yogyakarta, Indonesia. He is currently an Associate Professor at the Department of Computer Science and Electronics, Universitas Gadjah Mada, Indonesia. He is actively participating as a member of the societies as IEEE, ICROS. His research interests include digital image processing, pattern recognition, machine learning, computer vision, and software engineering. He has published many papers in reputable international journals indexed by Scopus in the fields of computer vision, image processing, and machine learning. He also served as a reviewer and editor in several international journals.