Analyzing the Performance of Approx-SMOTE Algorithm for Handling Imbalanced Big Data on Apache Spark

Kai Yun Ong
Selangor, Malaysia
kaiyun0904@gmail.com

I. Introduction

APACHE Spark is a popular open-source framework used for processing big data [15]. It includes machine learning library, Mllib, which offers a wide range of algorithms and tools for machine learning [12]. However, it has some limitations in terms of functionality compared to other machine learning frameworks. Therefore, many algorithms such as Synthetic Minority Over-sampling Technique (SMOTE) [3], which is based on the k-Nearest Neighbor (k-NN) algorithm, need to be adapted and integrated into the Spark ecosystem. This article [8] is worth to review as it presents a realistic solution to a common challenge in the field of machine learning which is the class imbalance problem. The solution provided is to implement the Approx-SMOTE algorithm [8], which is a parallel implementation of the SMOTE algorithm using an approximated version of k-Nearest Neighbor algorithm.

II. Problems statement

Imbalanced learning is a common issue encountered in classification datasets, where one class has significantly fewer instances than the other. This imbalance can lead to classifiers exhibiting a bias towards the majority class, resulting in misclassification of minority class instances [4, 7, 5]. Resampling the dataset to obtain a more balanced distribution of instances across classes is an effective approach to address this problem. While SMOTE [3] is one of the most popular resampling methods for addressing imbalanced learning, its computational intensity makes it infeasible for large-scale datasets in Big Data scenarios.

The hypothesis presented in the research paper is that a novel resampling method called Approx-SMOTE will outperform other techniques, such as SMOTE-BD [2], by reducing computing power while ensuring high scalability and maintaining classification performance. This article contributes to the field of imbalanced learning in Big Data scenarios by proposing the Approx-SMOTE algorithm, which is a resampling method that uses an approximated nearest neighbor search approach to address computational efficiency and scalability issues of existing methods.

The hypothesis is proven through experiments and evaluation of the performance of the Approx-SMOTE algorithm on different large datasets.

III. Related Work Discussion

TABLE I
A summary of the articles that are related to the reviewed article of Approx-SMOTE.

Article Title	Summary & How it relates to the reviewed article
SMOTE: Synthetic Minority Over-sampling Technique [3].	The paper discusses an approach for constructing classifiers from imbalanced datasets by combining over-sampling of the minority class and under-sampling of the majority class, resulting in better classifier performance. This approach is called Synthetic Minority Over-Sampling Technique (SMOTE). However, the donwside of SMOTE is that it uses k-NN to find the neighbourhoods of each instance, which can become computationally intensive when dealing with large datasets due to the quadratic time complexity. Hence, it is not feasible to use in big data senarios. As such, approx-SMOTE could be an alternative solution on top of SMOTE to make it scalable for large datasets.
An Investigation of Practical Approximate Nearest Neighbor Algorithms [10].	The paper proposes the use of approximate nearest neighbor searching algorithms for high dimensional perception areas such as computer vision, which could potentially be more efficient than Locality Sensitive Hashing (LSH) for certain applications. It introduces a new kind of metric tree that allows overlap and new approximate k-NN search algorithms on this structure. This approach could be a used in approx-SMOTE, which uses approximate k-NN for synthesising new examples in an imbalanced binary classification for large datasets.
Clustering Billions of Images with Large Scale Nearest Neighbor Search [11].	The paper discusses the challenges of performing computer vision, image processing and machine learning tasks on large scale image collections containing billions of images. The authors focus on the nearest neighbor search problem, which is a fundamental subproblem in many complex algorithms such as image clustering. They propose a scalable version of an approximate nearest neighbor search algorithm that can be used to find near duplicates among large scale image collections. The approximate k-NN approach could potentially be integrated into the approx-SMOTE algorithm to make it more scalable and efficient for large datasets.
SMOTE-BD: An Exact and Scalable Oversam-plingMethod for Imbalanced Classification in Big Data [2].	The paper addresses the challenges of imbalanced classification in the context of big data analytics and proposes a scalable preprocessing approach called SMOTE-BD, based on the SMOTE algorithm. However, it has limitations as it uses an iterative implementation of exact k-NN that requires significant amount of computational powers, which makes it impractical for solving real-world Big Data classification problems. In contrast, Approx-SMOTE uses approximate k-NN and is designed to be faster and more scalable than SMOTE-BD.

IV. Methodology

Fig. 1 shows the architecture of Approx-SMOTE. From the architecture, we can see that the Approx-SMOTE method utilizes the instance package and knn package. In the instance

TABLE II
SIX IMBALANCED DATASETS USED FOR ASSESSING CLASSIFICATION PERFORMANCE.

Dataset	# Instances	# Attributes	# maj/min	IR	Size (GB)
SUSY IR4	3389320	18	2712173/677147	4.00	1.23
SUSY IR16	2881796	18	2712173/169623	15.99	1.04
HIGGS IR4	7284166	28	5829123/1455043	4.00	3.94
HIGGS IR16	6194093	28	5829123/364970	15.97	3.26
HEPMASS IR4	6561364	28	5250124/1311240	4.00	3.77
HEPMASS IR16	5578586	28	5250124/328462	15.98	3.20

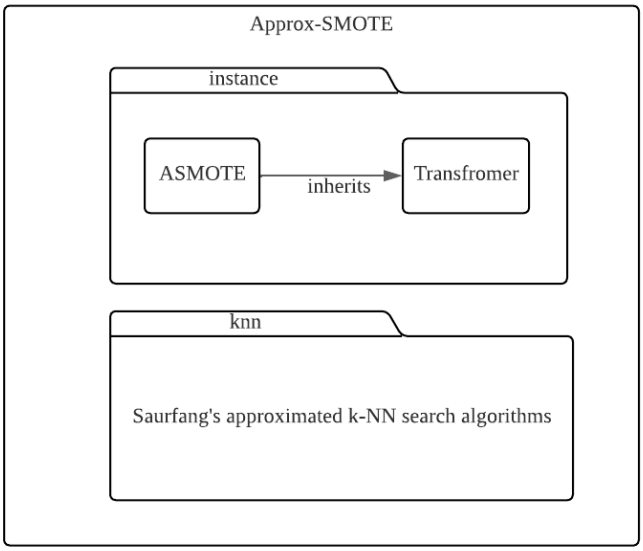

Fig. 1. Architecture of Approx-SMOTE.

package, the ASMOTE class inherits the Transformer class, whereas in the knn package Saurfang's approximated k-NN search algorithm is implemented. Approx-SMOTE offers flexibility in adjusting the parameters of Saurfang's approximated k-NN, such as the number of nearest neighbors, maximum distance between two neighbors, and the size of the top tree. This approximated k-NN method uses a hybrid spill tree approach to achieve high accuracy and search efficiency [8]. It is used for advanced clustering use cases and is optimized for efficient querying using search trees. The approximated k-NN model can be trained with different parameters to control the number of partitions as well as the accuracy and efficiency trade-offs. Although the implementation is approximate, it is adaptable to high dimensional dataset and requires sub-linear runtime which make it even scalable [6].

The experiment for Approx-SMOTE, as shown in Fig. 3, involved the validation of the algorithm through the use of cloud-based clusters. This approach enabled a comprehensive testing and evaluation of the algorithm's performance in a distributed computing environment. The objective is to prove that Approx-SMOTE can oversample data as effectively as SMOTE-BD, but in a faster and more scalable way. Fig. 2 shows how the oversampling works in general. The percOver represents the oversampling percentage. The higher the percentage, the more synthetic instances to be created in the minority class

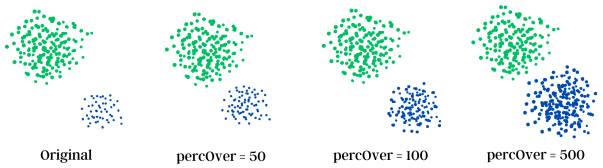

Fig. 2. Illustration on how Approx-SMOTE resamples a bivariate synthetic dataset consisting of 7,000 instances: 6,000 belongs to the majority class and 1,000 to the minority class.

(represented in blue).

The characteristics of datasets used for the experiment are displayed in Table II. IR represents the Imbalance Ratio.

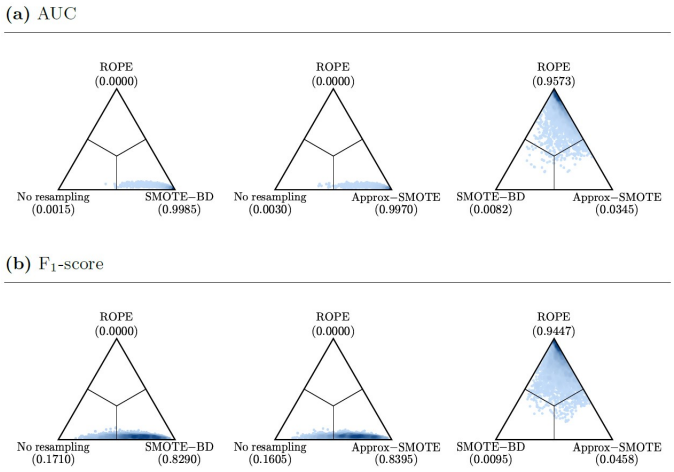

Fig. 4. Bayesian Hierarchical sign tests demonstrating the equivalence between SMOTE-BD and Approx-SMOTE [8].

V. ANALYSIS OF RESULT

Based on the result presented in Fig. 5, the dataset which has lower imbalance ratio (IR4) performs better than those with higher imbalance ratio (IR16). For example, the AUC and F1 score is higher in SUSY IR4 than SUSY IR16 regardless of the resampling algorithm used. In addition, the small standard deviation of the AUC and F1-score measurements indicates that the results are consistent across different runs of the experiments. For most of the datasets, both resampling techniques (SMOTE-BD and Approx-SMOTE) outperform the no resampling technique. The improvement is particularly significant

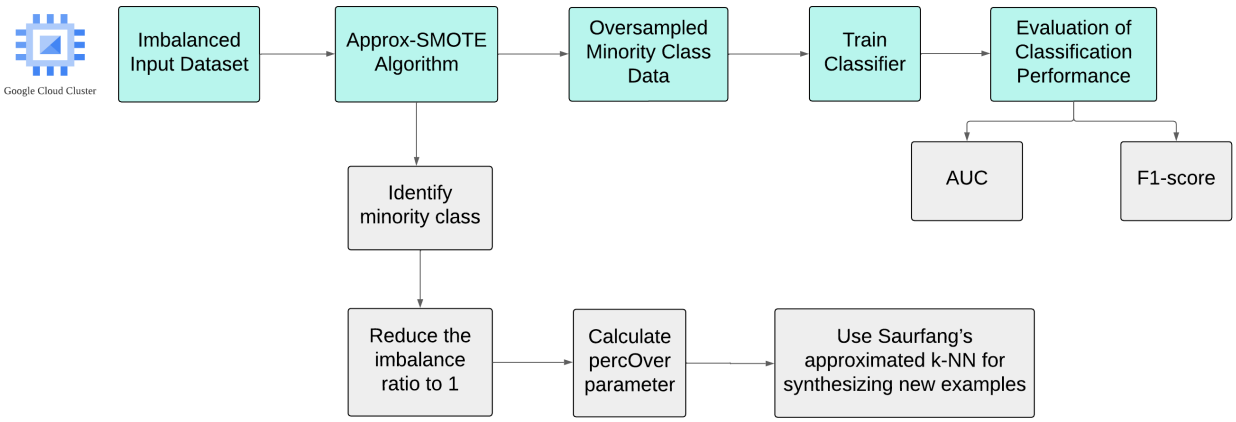

Fig. 3. Experiment's flow for Approx-SMOTE.

in terms of F1-score, where the resampling techniques outperform the no resampling technique in most of the datasets. For example, the F1-score of HIGGS IR4 dataset increases from 0.028 (no resampling) to 0.459 (Approx-SMOTE) as shown in Fig. 5. It can also be observed that the performance of SMOTE-BD and Appr-SMOTE in terms of AUC and F1-score is very similar in all of the experimented datasets, with only minor differences between the two techniques. This shows that both of the resampling methods performs equally well in the classification problem but better than no resampling.

To determine whether Approx-SMOTE outperforms other methods, Bayesian statistical tests in Fig. 4 were conducted and to determine whether the difference between two groups falls within the Region of Practical Equivalence (ROPE). The value of ROPE ranges from 0 (no difference) to 1 (significant difference). The higher the value of ROPE, the smaller the difference between the two groups. For AUC, when the no resampling technique is compared with the Approx-SMOTE, there is a probability of 99.70% that the Approx-SMOTE to perform better than the no resampling technique on a dataset. In addition, the ROPE of 0 shows that there is a significant difference between both methods. Similarly, SMOTE-BD performs significantly better than no resampling with a probability of 99.85%. On the other hand, when the SMOTE-BD is compared with the Approx-SMOTE, there is a probability of 95.73% that the two algorithms fall within the ROPE region. This suggests that the difference between the two algorithms is not practically significant and they may perform equally in real-world applications.

However, there is a huge difference in terms of the execution time and scalability between SMOTE-BD and Approx-SMOTE. As the cluster size increases from 2 workers (the smallest cluster) to 10 workers (the biggest cluster), as shown in Fig. 6, the execution time of Approx-SMOTE decreases whereas the execution time of SMOTE-BD increases disproportionately. More specifically, Approx-SMOTE is between 7.52 to 28.15 times faster than SMOTE-BD across all cluster sizes which can be calculated using the formula:

$$\frac{\text{Execution time of Approx-SMOTE in the current cluster}}{\text{Execution time of SMOTE-BD in the current cluster}} \quad (1)$$

The speed-up of each algorithm can also be computed using the smallest cluster configuration (2 workers) as a baseline as shown in Fig. 7. The formula to calculate the speed-up is shown below:

$$\frac{\text{Execution time of the algorithm in the smallest cluster}}{\text{Execution time of the algorithm in the current cluster}} \quad (2)$$

The speed-up comparison is a common practice in evaluating the performance of parallel algorithms, as it measures the relative improvement in performance achieved by parallelizing the algorithm with respect to a sequential implementation [1]. From Fig. 7, we can observe that as the number of workers increases, the speed up for Approx-SMOTE increases, whereas the speed up for SMOTE-BD decreases. This suggests that Approx-SMOTE scales well as more computational resources are added, while SMOTE-BD experiences scalability issues as the size of the clusters increases. This is an important finding because scalability is a critical aspect of big data processing. An algorithm that does not scale well can become a bottleneck in the processing pipeline. However, due to the fact that the experimental results are only based on a limited number of datasets, it may not hold true for all scenarios. This is a valid concern as the scalability of the algorithms also depends on various factors, such as network latency, processing power of individual workers, the size and complexity of data, the hardware and software environment, and specific implementation of the algorithm. These factors may vary across different cluster configurations and datasets. To further validate this findings, it is essential to test the algorithms on different datasets and cluster configurations, and compare the performance of these algorithms with other state-of-the-art algorithms for handling imbalanced big data.

VI. Critical Reasoning and Conclusion

While approximated k-NN for nearest neighbor search has been shown to outperform the traditional k-NN method which

Dataset	AUC			F_1-score		
	No resampling	SMOTE-BD	Appr-SMOTE	No resampling	SMOTE-BD	Appr-SMOTE
SUSY IR4	0.691 ± 0.002	0.771 ± 0.001	0.772 ± 0.001	0.542 ± 0.003	0.583 ± 0.003	0.589 ± 0.003
SUSY IR16	0.622 ± 0.009	0.771 ± 0.001	0.772 ± 0.001	0.380 ± 0.019	0.293 ± 0.003	0.300 ± 0.004
HIGGS IR4	0.507 ± 0.002	0.671 ± 0.002	0.678 ± 0.002	0.028 ± 0.008	0.452 ± 0.002	0.459 ± 0.002
HIGGS IR16	0.500 ± 0.000	0.660 ± 0.001	0.672 ± 0.002	0.000 ± 0.000	0.197 ± 0.001	0.202 ± 0.001
HEPMASS IR4	0.748 ± 0.004	0.821 ± 0.001	0.821 ± 0.001	0.655 ± 0.007	0.633 ± 0.003	0.630 ± 0.002
HEPMASS IR16	0.591 ± 0.014	0.822 ± 0.001	0.822 ± 0.001	0.306 ± 0.040	0.338 ± 0.004	0.329 ± 0.005

Fig. 5. Classification performance displayed in AUC and F1 Score. Best results are highlighted in black boxes, with bluer boxes indicating better performance. Standard deviation between cross-validation folds is reported as a ± value on the right [8].

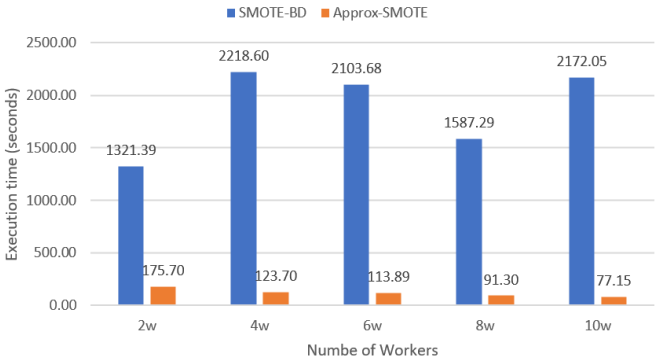

Fig. 6. Comparison of execution time between Approx-SMOTE and SMOTE-BD.

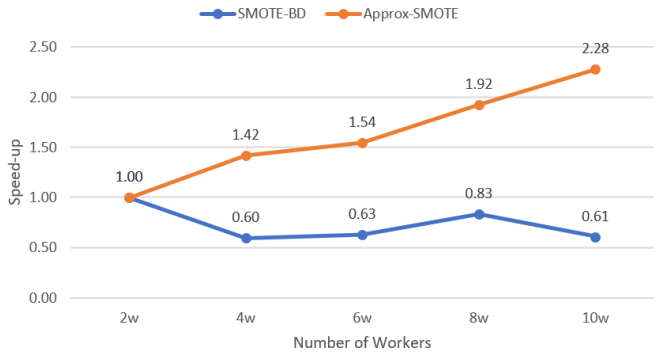

Fig. 7. Comparison of speed-up between Approx-SMOTE and SMOTE-BD.

uses exact nearest neighbour search, it raises the concern that this method relies on hybrid spill tree search, which requires significant amount of spaces. This means when the size of the datasets grows bigger, more spaces will be required to store the spill tree, which can lead to scalability issues. This is a major concern, particularly in high-dimensional datasets where the number of features is large [11]. One possible solution to address the space limitation in approximated k-NN is to use random projection as a pre-processing step to embed the dataset in a lower-dimensional subspace with minimal distortion on the pair-wise distances [10]. This technique is effective because it can significantly reduce the dimensionality of the dataset, which in turn reduces the computational cost of the approximated k-NN algorithm. In addition, multiple rounds of random projections can be implemented to ensure the accuracy of the algorithm. This approach has been investigated in several research studies and has demonstrated promising results in improving the scalability of approximated k-NN search.

Further research avenues exist to enhance the approximated k-NN search approach, such as exploring the use of hashing techniques to enhance the efficiency of the algorithm in identifying nearest neighbors within high-dimensional data spaces. By utilizing hashing, it may be possible to map high-dimensional data points into a lower-dimensional space, thereby reducing the number of distance calculations needed to identify the nearest neighbors. This area of research has shown promising results in various fields of machine learning and could potentially provide valuable insights into optimizing the performance of k-NN algorithms for large-scale datasets. One of the proposed algorithms is Grid Hashing Neighborhood (GHN) [13], which uses hashing and a virtual grid to select nearest neighbors directly in the neighborhood of a given observation, instead of searching through the entire dataset. This approach improves the time efficiency while maintaining the prediction quality. Indeed, it outperforms the original k-NN algorithm. Another proposed algorithm for approximate similarity search is locality-sensitive hashing (LSH) [13]. The study has shown that LSH outperforms traditional tree-based spatial data structures in high-dimensional spaces. The LSH algorithm can handle a large number of dimensions and data sizes and is suitable for real-time processing. However, there is still room for potential improvement of the LSH algorithm. This could involve exploring more systematic methods for creating data structures and investigating hybrid data structures obtained by merging tree-based and hashing-based approaches. Based on the existing researches about the hashing search algorithm, there is a possibility to embed the approach to Approx-SMOTE to synthesis new examples for minority class using approximated hashing nearest neighbour approach. Furthermore, several studies also show that setting thresholds or weighting the continuous output of a classifier may produce

superior results to data resampling, and can be implemented with any conventional classifier [9, 14].

In summary, the paper introduces a new oversampling method, called Approx-SMOTE, that is designed to address the imbalanced classification problem in Big Data. By incorporating an approximated k-nearest neighbor search approach into the SMOTE algorithm, Approx-SMOTE generates synthetic examples for the minority class in a more efficient and scalable manner than the existing SMOTE-BD implementation. Notably, the proposed method achieves a speed-up of up to 30 times when compared to SMOTE-BD in a 10-worker node cluster. Future research directions may include investigating the effectiveness of other fast approximate nearest neighbor search algorithms, such as hashing, within the context of SMOTE.

References

[1] Balkanski, E., Rubinstein, A., and Singer, Y. (2019). An exponential speedup in parallel running time for submodular maximization without loss in approximation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 283–302. SIAM.

[2] Basgall, M. J., Hasperué, W., Naiouf, M., Fernández, A., and Herrera, F. (2018). Smote-bd: An exact and scalable oversampling method for imbalanced classification in big data. In *VI Jornadas de Cloud Computing & Big Data (JCC&BD)(La Plata, 2018)*.

[3] Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357.

[4] Chawla, N. V., Japkowicz, N., and Kotcz, A. (2004). Special issue on learning from imbalanced data sets. *ACM SIGKDD explorations newsletter*, 6(1):1–6.

[5] Díez-Pastor, J. F., Rodríguez, J. J., García-Osorio, C. I., and Kuncheva, L. I. (2015). Diversity techniques improve the performance of the best imbalance learning ensembles. *Information Sciences*, 325:98–117.

[6] Fang, F. (2021). spark-knn: k-nearest neighbors algorithm on spark.

[7] He, H. and Garcia, E. A. (2009). Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9):1263–1284.

[8] Juez-Gil, M., Arnaiz-Gonzalez, A., Rodriguez, J. J., Lopez-Nozal, C., and Garcia-Osorio, C. (2021). Approx-smote: fast smote for big data on apache spark. *Neurocomputing*, 464:432–437.

[9] Krawczyk, B. and Woźniak, M. (2015). Cost-sensitive neural network with roc-based moving threshold for imbalanced classification. In *Intelligent Data Engineering and Automated Learning–IDEAL 2015: 16th International Conference, Wroclaw, Poland, October 14-16, 2015, Proceedings 16*, pages 45–52. Springer.

[10] Liu, T., Moore, A., Yang, K., and Gray, A. (2004). An investigation of practical approximate nearest neighbor algorithms. *Advances in neural information processing systems*, 17.

[11] Liu, T., Rosenberg, C., and Rowley, H. A. (2007). Clustering billions of images with large scale nearest neighbor search. In *2007 IEEE workshop on applications of computer vision (WACV'07)*, pages 28–28. IEEE.

[12] Meng, X., Bradley, J., Yavuz, B., Sparks, E., Venkataraman, S., Liu, D., Freeman, J., Tsai, D., Amde, M., Owen, S., et al. (2016). Mllib: Machine learning in apache spark. *The Journal of Machine Learning Research*, 17(1):1235–1241.

[13] Tchaye-Kondi, J., Zhai, Y., and Zhu, L. (2020). A new hashing based nearest neighbors selection technique for big datasets. *arXiv preprint arXiv:2004.02290*.

[14] Yu, H., Sun, C., Yang, X., Yang, W., Shen, J., and Qi, Y. (2016). Odoc-elm: Optimal decision outputs compensation-based extreme learning machine for classifying imbalanced data. *Knowledge-Based Systems*, 92:55–70.

[15] Zaharia, M., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., Venkataraman, S., Franklin, M. J., et al. (2016). Apache spark: a unified engine for big data processing. *Communications of the ACM*, 59(11):56–65.