Application of Evolutionary Computation for High Velocity Streaming Data

Wenbo Li 11710710, Jiajun Yu 11711102

Abstract

Credit card fraud became a significant issue around the world. Our topic is to solve the major problems of fraud detection, including imbalanced data set and concept drift. In these weeks, we tried to improve a windowing algorithm named FLORA, and we added an oversampling method called SMOTE. This report discusses how we improved the FLORA algorithm, and introduces the SMOTE algorithm simply. It also shows the result of each experiment. Besides, this report also states what we will do in the future.

I. Introduction

According to The Nilson Report in 2019, for every 100 dollars of the total volume of the card-based payment systems, the fraud loss would be 6.86 cents. [1] Besides, card fraud reached 27.85 billion dollars.[1] Along with the development of Science and Technology, human's lives became more and more convenient. The pattern of payment is also being changed by the advanced technology. Credit card payment predominates in the patterns of the payment nowadays. However, this kind of payment does not need a face-to-face trade, which makes fraud more likely to happen. Credit card fraud became an increasingly serious problem nowadays.

The traditional machine learning methods are mostly concentrating on a given and well defined data set(sufficient and representative). Theoretically, these methods are able to deal with the data efficiently and accurately. However, in real world scenarios, the data can rarely fit in such restricted models. The class distribution may be very awful, extreme imbalanced class distribution are common situations. What's more, the data often comes in continuously in reality. In this kind of streaming data, changes of the data concept or distribution may cause concept drift.

To deal with the concept drift, we looked into several algorithms and found that the windowing algorithm can help. FLORA algorithm is one of the windowing algorithms. [2][3] FLORA aims to make the window be able to adapt to the streaming data automatically. We found this algorithm easier to improve and implement, and the space complexity is lower than the ADWIN algorithm. [4] Therefore, we tried to use the FLORA algorithm and found that the recall rate is higher than of our experiment last semester.

For these weeks, we repeated the experiment several times and tried to improve the FLORA algorithm. We first modified the judging criteria for adjusting the window size, and we also tried to train the fraud entry twice, which we thought might be a way to oversample. After that, we also added a SMOTE algorithm to oversample the fraud data. [5] However, we found that the SMOTE algorithm performed badly.

The report would be divided into six sections. The following section would be the literature review, which simply introduces several kinds of algorithms to handle concept drift and imbalance. The third section would be the introduction of the FLORA algorithms. In Section 4 we would show our experiment results. Additionally, Section 5 would be a comparison between our experiment and some open source solutions. The last section would show the evaluation of our experiments this semester and would discuss future work.

II. Literature Review

A. Dealing with concept drift

T. Ryan Hoens et al stated that to handle the concept drift, there would be three types: adaptive base learners, learners that amends the training data set, and an integral approach.[6]

1) Adaptive base learner: The adaptive base learner is considered the easiest way to dispose of the concept drift. The major aim is to make the learning model be able to fit the training data set whose concept is different from the concept that the model has learned.

For the decision tree model, P. Domingos and G. Hulten argued that the Hoeffding bound would be a good way to deal with the streaming data.[7] This is an algorithm that uses the error to detect and manage the concept drift of a decision tree model, called Very Fast Decision Tree(hereinafter referred to as VFDT). Later, Hulter et al optimized the VFDT and called it CVFDT.[8] This CVFDT used a sliding window algorithm, which can help the model forget the discarded training examples.

Moreover, A. Bifet and R. Gavalda raised a Hoeffding Window Tree(hereinafter referred to as HWT) to optimize the Hoeffding Tree.[9] This HWT algorithm does not need to speculate the speed and the frequency with which the streaming data changes, which means that there would be fewer parameters to be defined by users. This is also a

sliding-window based algorithm, and the behavior is better than the CVFDT.

Besides, there was another kind of decision tree method called option tree, in which there would be some option nodes, to which have several different paths from the root, added into the tree.[10][11] Later in 2007, B. Pfahringer et al invented an algorithm using the Hoeffding Tree as a basic algorithm and added some option nodes, called Hoeffding Option Tree(hereinafter referred to HOT).[12] Bifet et al extended the HOT method as Adaptive HOT(AHOT), of which the nodes would be provided by some parameters estimating the shifting of the streaming data.[13]

2) Modification to the training set: To alter the training data set, there are two major approaches: by window or weight.

For window approaches, we can first determine a steadfast window size, which represents the total number of the "new" training examples, that is, the latest streaming data. The main problem of the artless model is the difficulty of estimating a suitable size. Subsequently, G. Widmer and M. Kubat designed a FLORA3 method, which altered from the FLORA algorithm that first introduced by Kubat. [2][3] FLORA stands for Floating Approximation, which is an algorithm that can be used for everlasting learning and updating the knowledge base. When some new data came, the oldest data would be forgotten, and the knowledge base would be changed. Subsequently, the FLORA3 algorithm was developed by Widmer and Kubat, which can automatically regulate the window size to adapt the knowledge base.

Later, several new window algorithms were designed. Klinkenberg and Joachims devised an algorithm which is Support Vector Machine(SVM) based.[14] They use a $\xi\alpha$-estimator as a parameter to evaluate the error rate of the window. The window size would also be changed to minimize the $\xi\alpha$-estimator.

Gama et al produced another method for the decision tree, neural network, and perceptron model.[15] They mainly stated that a probability of predicting false p_i at point i, and the standard deviation $s_i = \sqrt{\frac{p_i(1-p_i)}{i}}$, which are used to detect concept drift. There would be a warning threshold and a detecting threshold, when $p_i + s_i$ is greater than the thresholds, the method would warn or detect the drift.

Bifet and Gavalda stated an adaptive window method(ADWIN2) which determines the window size by some sub-windows.[4] This method is optimized from the ADWIN method which has to maintain two large sub-windows. The sub-windows are "large enough", and when they are recognized as two "distinct enough" sub-windows, the concept drift would be detected. Still, the ADWIN method needs a large space and it would take a long time to check all pairs of the "large enough" sub-windows. Hence, Bifet and Gavalda produced the ADWIN2 approach to reduce the time and space complexity. They use a data structure that can store a window whose size is W in a space cost $O(logW)$.

The weighting methods seem to be used less frequently compared to the windowing approaches. Alippi and Roveri introduced a method that is k-Nearest Neighbor(KNN) based.[16] They assign weights that depend on the congruence with the current concept to each example. When new data came, the weight of the old data would decrease, and thus, contribute less to the result.

3) Ensemble methods: The ensemble techniques are commonly used now. We mainly consider bagging or boosting based ensemble learning methods. The boosting algorithm has been proved to have a better performance than the bagging algorithm. Last semester we tried to use the XGBoost algorithm and found that it can provide a reasonable result.

Polikar et al provided a Learn++ method that is an incremental learning method based on AdaBoost.[17] Also, Chu and Zaniolo introduced an adaptive boosting ensemble(ABE).[18] When it detected a concept drift, it would notify the ensemble and the ensemble would be dropped by the ABE.The ABE would then learn again. Besides, Bifer et al devised an ensemble method called ADWIN bagging which is improved from the ADWIN2 method.

B. Handling the issue of imbalance

To dispose the imbalanced data set, we mainly have two methods: oversampling and undersampling. For the oversampling method, we repeat training the data which is fraudulent taking fraud detection as an example. The oversampling methods include Borderline-SMOTE[20], ADASYN[21], and MWMOTE[22]. Also, there are undersampling techniques(i.e. shrinking the majority class) like Tomek links[23], One-sided selection[24], Neighborhood cleaning rule[25], etc.

C. Dealing with concept drift and imbalanced data set simultaneously

As we mentioned earlier, fraud detection is hard because we have to handle the streaming data and the imbalanced data set together. In last semester we tried to use a simple sliding window method and manually divided the data set to deal with the concept drift and imbalanced data set at the same time. However, those were not efficient and it would be hard to divide the data set by the users of the program. Therefore, this semester we read some literature and found some methods to deal with the two problems simultaneously. One of the latest methods is Learn++.NIE algorithm.

G. Ditzer and R. Polikar introduced a Learn++.NIE algorithm which can be used to solve the two problems together.[26][27] This can be used to learn neural network classifiers. In their previous work, they produced a Learn++.NSE method that can be used to detect concept drift in a simple model. The Learn++.NIE, in which NIE stands for non-stationary and imbalanced environment, is with a bagging algorithm. When creating the bag, it would use under-sampling to make the data set less imbalanced.

III. Algorithm Introduction

After looking at all of the algorithms, our first thought is to continue to work on windowing algorithms. Windowing algorithms include FLORA, ADWIN, etc. ADWIN needs to compare the two windows that are big enough, which I think may have a high space complexity. Therefore, we decided to work on the FLORA algorithm. This algorithm aims to change the window size automatically. This algorithm may have to take a long time because while learning or forgetting, it needs to search for the description items base in each iteration. To make the algorithm more effective for fraud detection, we make some changes.

The algorithm needs three bases: $ADES$ to store the positive classification instances, $PDES$ to store the "potential" classification instances (will be introduced later), and $NDES$ to store the negative classification instances. The items in $ADES$ would be the instances and the ap values are the counters of each item. Similarly, items in $NDES$ would be the instances and the nn values are the counters of each item. $PDES$, however, would store the instances and two counters, in which $pneg$ record the counter of the item removed from $NDES$, and in which $ppos$ record the counter of the item removed from $ADES$. These three bases should have been composed of Disjunction Normal Forms(DNF), but the numbers in the data set we use have high precisions, so it's hard to convert them into DNFs. Therefore, we store those instances on these bases. Also, because the data set has no repetitions, we don't have to check the three bases when we add instances to the window.

The algorithm also has three thresholds: lc that stands for the low coverage threshold of the positive classifications in $ADES$, hc that stands for the high coverage threshold of the positive classifications in $ADES$, and $pacc$ that stands for the acceptable accuracy. However, considering fraud detection, the data set is quite imbalanced. The data set we are going to handle contains around 0.17% frauds, that is, the accuracy is unpersuasive because if we consider all instances as non-frauds, we can get accuracy as 99.83%, which is meaningless. Therefore, we added another threshold prec which stands for the acceptable recall rate.

The algorithm can be divided into three parts: learning, forgetting, and adjusting the window.

A. Learning

Learning is the process when a new instance is being added into the window. If the instance is positive, we would store it into $ADES$, and then check if $PDES$ and $NDES$ have the same instance. If $PDES$ has the same instance, we add one to the counter $ppos$ of this instance, and if $NDES$ has the same instance, we should minus one from the counter nn of this instance, and if nn equals to 0, the instance should be popped out from the $NDES$.

If the instance is negative, we would store it into $NDES$, and then check if $PDES$ and $ADES$ have the same instance. If $PDES$ has the same instance, we add one to the counter $pneg$ of this instance, and if $ADES$ have the same instance, we should minus one from the counter ap of this instance, and if ap equal to 0, the instance should be popped out from the $ADES$.

B. Forgetting

Forgetting is the process when a new instance is being removed from the window. If the instance is positive, then we remove it from $ADES$, and we let the counter $ppos$ of this instance in $PDES$ minus one, and if $ppos$ equals to 0 it would be removed from $PDES$ and then stored into $NDES$ with a counter nn as $pneg$.

Otherwise, if the instance is negative, then we remove it from $NDES$, and we let the counter $pneg$ of this instance in $PDES$ minus one, and if $pneg$ equals to 0 it would be removed from $PDES$ and then stored into $ADES$ with a counter ap as $ppos$.

C. Adjusting window

This is the core method that is used to adjust the window size automatically. There are three kinds of conditions. One of them is that when the coverage of the positive classification in $ADES$ is too low (lower than lc) or the accuracy is too low (lower than $pacc$), and $PDES$ is not an empty set; another is that when the coverage of the positive classification in $ADES$ is extremely high, i.e., higher than twice of hc, and the accuracy is higher than $pacc$; the third of them is that when the coverage is relatively high, i.e. higher than hc but lower than twice of hc, and the accuracy is higher than $pacc$. All of the three conditions would lead to a reduction of the window size. According to our topic, the data set of fraud detection would be imbalanced, and recall rate would be a more convincing metric than accuracy. Therefore, we consider both of the two metrics.

D. Experiment results

For these weeks, we have done several experiments to test and improve the FLORA algorithm. First, we retried

the experiment last time and change some of the parameters. We set the recall threshold as 0.85 and accuracy threshold as 0.995, and then we got a recall rate as 87%, which was the highest recall rate that we have ever got. Later we thought about changing the conditions. To do the oversampling method, we tried to repeat training the frauds to increase the weight of the frauds. According to the results of the experiments that we have done before and the result from Kaggle, we found that along with the increasing of the recall rate, the accuracy would be likely to decrease, because the model might misclassify more legitimate behaviors as frauds. Therefore, we tried to find a balance between accuracy and recall. We then changed the boolean value "metric" into "int", and store three kinds of situations: 0 for bad behavior, which would decrease the window size by 20%; 1 for acceptable behavior, which would forget one item or two items (which depends on the proportion of the frauds in $ADES$); 2 for too high recall, which would skip the forget step and therefore add the window size by 1. After the modification, we run the program with different groups of data several times, but the recall rate and the accuracy could hardly reach 80%.

Later, we tried to use the SMOTE algorithm to oversample the data set, but we found that the data produced by SMOTE had low quality. To avoid the out-of-order of the data timestamps, we keep the "Time" value in x and put (x, y) as the input of the SMOTE model in an Imbalanced-learn package. Then we sort the entries by the "Time" value and later drop the "Time" value in case "Time" would influence the training result. If we decrease the desired ratio, both recall rate and accuracy would increase. Nonetheless, the recall rate was still under 80%, and the accuracy was also low.

For all experiments, we made a table to record the experiment results and parameters. Additionally, for all experiments with the SMOTE algorithm, we made a graph to show the changing tendency of the metrics. The table and the figure are shown in the appendix.

IV. Comparison with others

We also have some comparisons between our model and some of the solutions on Kaggle. First, we will talk about the most popular solution for Kaggle. In this case, the author used a random under-sampling technique to deal with the imbalanced dataset. He randomly picked 492 non-fraud cases and 492 fraud cases for training. The prediction precision with the decision-tree classifier finally achieved 91.7% on the under-sampling dataset. We reproduced his model last semester, and the recall rate on the original dataset was approximately 72.12%. This author also tried SOMTE oversampling technique, the recall rate was approximately 86%, which was quite satisfying. But the prediction precision of fraud cases was only about 10%. In this case, the model wrongly classified too many non-fraud cases as frauds, which was also unacceptable.

In the second popular solution, the author also used SMOTE for resampling. He chose logistic regression as the classifier. His model can achieve around 91% recall rate on the over-sampling dataset and 93.2% recall rate on the under-sampling test set. However, when the author tested the model with the original dataset, the result was not satisfying. Although the recall rate was still around 91%, this model classified 13% non-fraud cases as fraud cases. Because the fraud cases only make up less than 1% in practice, over ten percent of the transactions will be classified wrongly. So this case is also unacceptable.

There are also some other solutions on Kaggle. One of them used semi-supervised learning and AutoEncoders on the under-sampling dataset, but he didn't test it on the original dataset. Another solution used a neural network to classify, but the author didn't show the recall rate, only gave the result of the accuracy, which was 98.59%.

In conclusion, our model performs relatively well compared with these models. We didn't show the figure because some of those authors only give the best metric among all metrics, and also some Kaggle solutions are tested on different data sets.

V. Conclusion and Future Work

In conclusion, our model performs well compared with several models selected from Kaggle. The best thing for our model is that we balance the accuracy and the recall relatively well, and we didn't drop any useful information. Most of the models presented by other people used the undersampling data set, which would drop too much useful information, and might ignore time stamps.

On the other hand, the FLORA algorithm still needs to be optimized. One thing is that the speed is too low because for each training datum, the window size would be judged and the model would be retrained. Another is that it is difficult to transform the data forms to DNFs. Furthermore, the oversampling method should be changed or invented for fraud detection problems, for the Euclidean distance may bring a lot of problems when generating the data in the minority class. For example, the order of magnitude of one column may have a far cry from the order of magnitude of another column. This would cause a problem when finding the nearest neighbor of entry because the column with the greatest order of magnitude would dominate the influence to the Euclidean distance.

In this semester, we tested an existing algorithm to learn the method to handle the concept drift. We also tested several oversampling algorithms to deal with the imbalanced data set, although the result of it is unacceptable. We got

more familiar with the windowing algorithms, and we understood the oversampling and undersampling methods more clearly and deeply. For our next step, we would try another algorithm to deal with the concept drift, and then add an appropriate algorithm to handle the imbalanced data set. If we can find an algorithm that has a low time complexity, we can adjust the parameters of it by evolutionary computation methods such as climbing hills or simulated annealing.

REFERENCES

[1] The Nilson Report, "Card Fraud Losses Reach $27.85 Billion", 2019.

[2] Kubat, M.: Floating approximation in time-varying knowledge bases. PRL 10(4), 223–227 (1989)

[3] Widmer, G., Kubat, M.: Effective learning in dynamic environments by explicit context tracking. In: ECML, pp. 227–243. Springer, Berlin (1993)

[4] Bifet, A., Gavalda, R.: Learning from time-changing data with adaptive windowing. In: SDM, pp. 443–448 (Citeseer) (2007)

[5] N. Chawla, K. Bowyer, L. Hall and W. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique", Journal of Artificial Intelligence Research, vol. 16, pp. 321-357, 2002.

[6] Hoens, T.R., Polikar, R. and Chawla, N.V. Learning from streaming data with concept drift and imbalance: an overview. Prog Artif Intell 1, 89–101 (2012). https://doi.org/10.1007/s13748-011-0008-0

[7] Domingos, P., Hulten, G.: Mining high-speed data streams. In: KDD, pp. 71–80. ACM, New York (2000)

[8] Hulten, G., Spencer, L., Domingos, P.: Mining time-changing data streams. In: KDD, pp. 97–106. ACM, New York (2001)

[9] Bifet, A., Gavalda, R.: Adaptive learning from evolving data streams. In: IDA, pp. 249–260 (2009)

[10] Buntine, W.: Learning classification trees. Stat. Comput. 2(2),63–73 (1992)

[11] Kohavi, R., Kunz, C.: Option decision trees with majority votes. In: ICML, pp. 161–169. Morgan Kaufmann, Menlo Park (1997)

[12] Pfahringer, B., Holmes, G., Kirkby, R.: New options for hoeffding trees. In: AAI, pp. 90–99 (2007)

[13] Bifet, A., Holmes, G., Pfahringer, B., Kirkby, R., Gavaldà, R.: New ensemble methods for evolving data streams. In: KDD, pp. 139–148. ACM, New York (2009)

[14] Klinkenberg, R., Joachims, T.:Detecting concept drift with support vector machines. In: ICML (Citeseer) (2000)

[15] Gama, J., Medas, P., Castillo, G., Rodrigues, P.: Learning with drift detection. In: AAI, pp. 66–112 (2004)

[16] Alippi, C., Boracchi, G., Roveri, M.: Just in time classifiers: managing the slowdrift case. In: IJCNN, pp. 114–120. IEEE, New York (2009). doi:10.1109/IJCNN.2009.5178799

[17] Polikar, R., Upda, L., Upda, S.S., Honavar, V.: Learn++: an incremental learning algorithm for supervised neural networks. In:SMC Part C, pp. 497–508 (2001)

[18] Chu, F., Zaniolo, C.: Fast and light boosting for adaptive mining of data streams. In: PAKDD, pp. 282–292 (2004)

[19] Bifet, A., Holmes, G., Pfahringer, B., Kirkby, R., Gavaldà, R.: New ensemble methods for evolving data streams. In: KDD, pp. 139–148. ACM, New York (2009)

[20] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-smote: A new over-sampling method in imbalanced data sets learning," in International Conference on Intelligent Computing (ICIC), 2005, pp. 878–887.

[21] H. He, Y. Bai, E. A. Garcia, and S. Li, "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," 2008, pp. 1322–1328.

[22] S. Barua, M. M. Islam, X. Yao, and K. Murase, "MWMOTE - majority weighted minority oversampling technique for imbalanced data set learning," IEEE Transactions on Knowledge and Data Engineering, vol. 26, no. 2, pp. 405–425, 2014, an oversampling method by generating new examples.

[23] I. Tomek, "Two modifications of CNN," IEEE Transactions on Systems, Man and Cybernetics, vol. 6, no. 11, pp. 769–772, 1976.

[24] M. Kubat, R. Holte, and S. Matwin, "Learning when negative examples abound," in 9th European Conference on Machine Learning Prague, vol. 1224, 1997, pp. 146–153.

[25] L. Jorma, "Improving identification of difficult small classes by balancing class distribution," in 8th Conference on Artificial Intelligence in Medicine in Europe, AIME 2001, vol. 2101, 2001, pp. 63–66.

[26] Ditzler, G., Polikar, R.: An incremental learning framework for concept drift and class imbalance. In: IJCNN. IEEE, New York(2010)

[27] Ditzler, G., Polikar, R., Chawla, N.V.: An incremental learning algorithm for nonstationary environments and class imbalance. In: ICPR. IEEE, New York (2010)

APPENDIX

id	pacc	prec	SMOTE	modified FLORA	train size	test set accuracy	test set recall	AUC score of ROC curve
1	0.995	0.8	NONE	NO	31017	N/A	N/A	not stored
2	0.995	0.8	NONE	NO	227844	0.978688	0.84	not stored
3	0.995	0.8	NONE	NO	227844	0.978617	0.826667	not stored
4	0.98	0.8	NONE	NO	227844	0.978775	0.84	not stored
5	0.98	0.8	NONE	NO	227844	0.977002	0.866667	not stored
6	0.98	0.8	NONE	YES	227844	0.976019	0.8	not stored
7	0.985	0.83	NONE	YES	227844	0.986412	0.786667	not stored
8	0.99	0.85	NONE	YES	227844	0.986868	0.813333	not stored
9	0.99	0.93	NONE	YES	227844	0.977845	0.786667	0.882382
10	0.98	0.85	NONE	YES	227844	0.986465	0.8	not stored
11	0.99	0.83	0.25	YES	284285	0.998122	0.733333	0.865902
12	0.99	0.83	0.2	YES	272912	0.998841	0.413333	0.706473
13	0.99	0.83	0.1	YES	250170	0.995857	0.773333	0.884742
14	0.99	0.83	0.05	YES	238799	0.991415	0.8	0.895834
15	0.99	0.83	0.05	NO	238799	0.990151	0.733333	0.861911

TABLE I

EXPERIMENT RESULTS

Notice that the first experiment was terminated manually because of the low speed. And after optimizing the program, we then got the 1.0 version, and we used this program with $pacc$ = 0.995 and $prec$ = 0.8 and got two results (id 2 and 3), which are the results shown in our progress report. The result with id as 4 is the best result we have ever made, which was done after the mid-term presentation. Later, we made some changes to the judging criteria, which we discussed in the previous part, to balance between the recall and the accuracy. However, from results 5 to 10, we learned that the modified

FLORA performed worse than the 1.0 version. Then we added the SMOTE algorithm to oversample the data set, and from results 11 to 15 we could learn that the SMOTE algorithm performed badly. From the 12th experiment, we added a parameter to restrict the minimum proportion of the frauds in the window. No matter whether we used the changed condition and whether we restricted the minimum proportion of the frauds, the recall and accuracy rate of the test set is quite low.

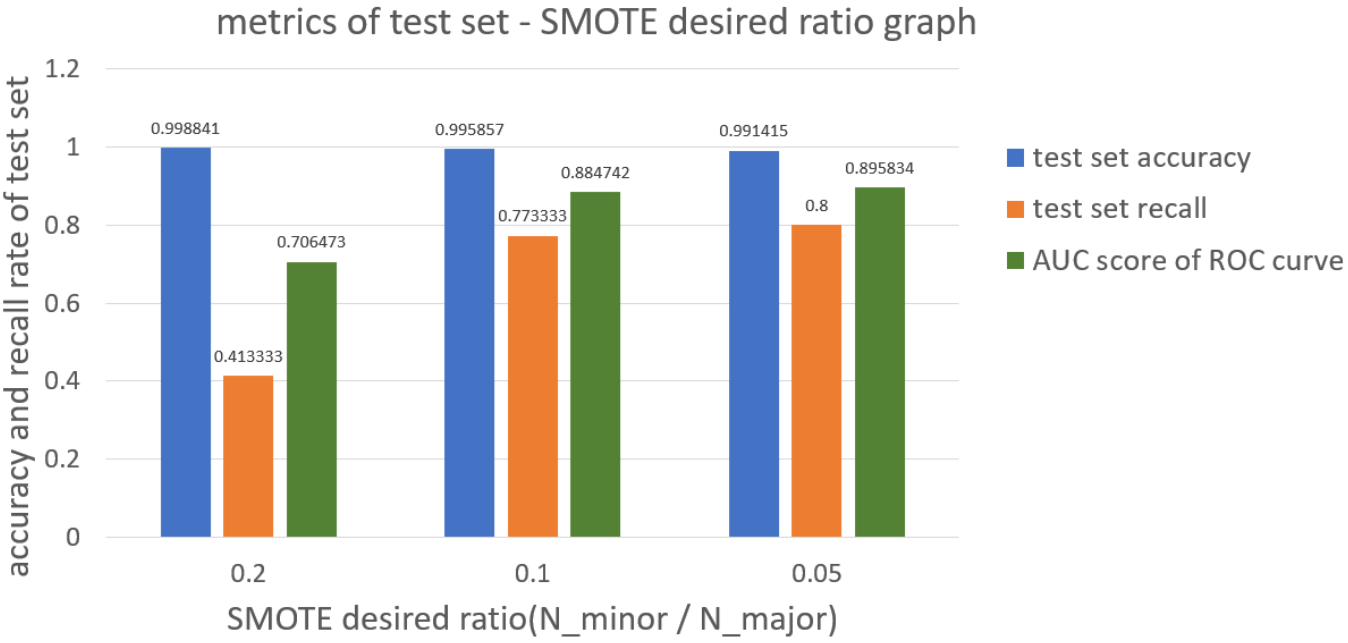

Fig. 1. Comparison of different desired ratio of SMOTE algorithm using flora version 2.1

From the figure, we can learn that the recall rate increases while the desired ratio of the SMOTE algorithm decreases. This proves that the SMOTE algorithm would generate too much bad data when the desired ratio is too high. Although the accuracy decreases a little bit when the desired ratio decreases, it doesn't influence the performance too much.