Robust Machine Learning with Imbalanced Data: Expanding Mixup

Andriamarolahy Rabetokotany
(andriamarolahy.rabetokotany@aims-senegal.org)
African Institute for Mathematical Sciences (AIMS)
Senegal

Supervised by: Dr. Vukosi Marivate

April 27, 2022

Submitted in Partial Fulfillment of a Masters II at AIMS

Abstract

Imbalanced data is part of the biggest issue in Machine Learning. Learning from such imbalanced data is called imbalanced learning which is a subfield of Machine Learning thoroughly studied. $SMOTE$ is one of the best techniques to approach this issue. However $SMOTE$ exhibits poor prediction towards minority class when data is highly skewed and just outside the training data. This relates to the fact that we use Empirical Risk Minimization (ERM) technique to train the model. But from a prominent study on this topic, some researchers were able to create an algorithm that can fix the drawbacks of ERM. This algorithm is called mixup. In this work we tried to combine the advantages of both $SMOTE$ and mixup techniques. The experimental results show that applying the combination of $SMOTE$ and mixup exhibits an high precision of prediction with binomial classification on imbalanced data.

Declaration

I, the undersigned, hereby declare that the work contained in this essay is my original work, and that any work done by others or by myself previously has been acknowledged and referenced accordingly.

[redacted]

Andriamarolahy Rabetokotany, April 27, 2022

List of plots

Tables

Contents

1. Introduction

In recent years, more and more data is being generated in real world situations. Accordingly, Machine Learning becomes one of the focus of the scientific community as it provides powerful tools to extract value from such data by giving computers the ability to learn [1] and improve from experience. With this increasing amounts of data becoming avalaible, there is a good reason to believe that *robust Machine Learning* will become more and more important as a necessary ingredient for technologies progress.

In contrast, several problem of Machine Learning applied to real world problems are posed. One of the most important among them is Imbalanced Learning problem, which is the ability of Imbalanced Data to significantly compromise the performance of most standard learning algorithms because they assume or expect balanced class distributions. Thus, these algorithms will be biased towards the majority class when presented with highly imbalanced datasets [3]. So, to overcome such problem one must design an intelligent algorithm that is able to capture all forms of that issue from imbalanced data. In fact, a lot of algorithms have been thouroughly developed. However, further study of this topic needs to be adressed. By doing so, deeper analysis of the structure and distribution of the imbalanced data should be studied in order to choose the appropriate method to use for a specific imbalanced learning problem.

The purpose of this work is to provide a comparative study of the current understanding of the imbalanced learning problem. By doing so, chapter 2 point out some literature survey about the state-of-the-art imbalanced learning. Then, chapter 3 covers some background study of Machine Learning. Next, chapter 4 focus on some experimentations by implemeting some techniques used in imbalanced learning. After that, chapter 5 looks at discussing the results of the previous experiment. Finally, in chapter 6, we give some conclusion and some perspectives.

2. Some definitions and literature survey of imbalanced learning

To be consistent, definitions of different useful terms, that we are going to see frequently, are defined in the following sections.

2.1 Some definitions

2.1.1 Machine learning. Machine Learning is a subfield of computer science. Its goal is to give computers the ability to learn by providing algorithms that can learn and make predictions on data [1]. Machine learning can be classified into supervised and unsupervised learning. By definition, in supervised machine learning, the machine learns with help of a labeled training data. Similarly unsupervised learning is the process of grouping data into similar categories.

2.1.2 Imbalanced data. Imbalanced data is a data set that exhibits a significant unequal distribution between its classes. The dominant groups in the data are called *majority classes* or *negative classes* [2]. However, the underrepresented groups in the data are called *minority classes* or *positive classes* [2].

2.1.3 Imbalanced learning. Imbalanced learning is another way to call the process of learning from imbalanced data [3]. In general, there are two methods for imbalanced learning: data level and algorithm level methods. For the data level methods, the goal is to balance the distributions between classes by doing preliminary data analysis. Similarly, the data level methods modify existing learning algorithms to alleviate the bias towards the majority classes.

The next section provides some survey in the state-of-the-art research in imbalanced learning that we can use as benchmarks.

2.2 Literature survey of imbalanced learning

Imbalanced data has long been recognized as a fundamental problem in machine learning. As far back as 2009, Haibo He and Edwardo A. Garcia provided a comprehensive review of the development of research in learning from imbalanced data. Their focus was to provide a critical review of the nature of the problem, the state-of-the-art technologies, and the current assessment metrics used to evaluate learning performance under the imbalanced learning scenario. Furthermore, in order to stimulate future research in this field, they also highlighted the major opportunities and challenges, as well as potential important research directions for learning from imbalanced data. Many researchers have focused on Haibo He et al' s work and tried to give directions on the future research in this topic. For example, Sofia Visa and Anca Ralescu talked about the recent progress in the field of learning of imbalanced data. It reviews approaches adopted for this problem and it identifies challenges and points out future directions in this relatively new field. Bartosz K. (2016) opened the discussion from skewed distribution of binary tasks to different area of research in topic of learning where the imbalanced data is the problem. He discussed about different issues and open challenges that need to be addressed to further develop the field of imbalanced learning.

A variety of specific techniques have been suggested to address the problem of imbalanced learning.

For example, Rushi Longadge, Snehlata S. Dongre and Latesh Malik (2013) studied each approach that define different methods available for classification of imbalance data set. They gave the right direction for research in class imbalance problem. Nathalie Japkowicz demonstrated experimentally that, at least in the case of connectionist systems, class imbalances hinder the performance of standard classifiers and compares the performance of several approaches previously proposed to deal with the problem. Guillaume Lemaître, Fernando Nogueira and Christos K. Aridas(2017) created a python toolbox that aims to provides a wide range of methods to cope with the problem of imbalanced learning in which the authors implemented sampling and ensemble learning techniques. As part of the sampling techniques implemented in this previous python tools, SMOTE is one of the successful approach for imbalanced learning. Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall and W. Philip Kegelmeyer (2002) talked about SMOTE which showed that a combination of their method of over-sampling the minority class and under-sampling the majority class can achieve better classifier performance than other techniques. Their methods of over-sampling the minority class involves creating synthetic minority class examples. Experiments are performed using C4.5, Ripper and a Naive Bayes classifier. The method was evaluated using the area under the Receiver Operating Characteristic curve and the ROC convex hull strategy. Marking the 15-year Anniversary of SMOTE, Alberto Fernández, Salvador Garciá, Francisco Herrera and Nitesh V. Chawla (2018) discussed the current state of affairs with SMOTE, its applications, and also identify the next set of challenges to extend SMOTE for Big Data problems.

Despite the success of the approaches cited above, Hongyi Zhang,Moustapha Cisse, Yann N. Dauphin and David Lopez-Paz (2018) raised problems in large deep neural networks: 1) undesirable behaviors such as memorization and 2) sensitivity to adversarial examples, which are trained with empirical risk minimization (ERM)technique. They proposed mixup, a simple learning principle to alleviate the undesirable behaviors such as memorization and sensitivity to adversarial examples . They trained a neural network on convex combinations of pairs of examples and their labels. By doing so, they regularize the neural network to favor simple linear behavior in-between training examples. Their experiments on the ImageNet-2012, CIFAR-10, CIFAR-100, Google commands and UCI datasets show that mixup improves the generalization of state-of-the-art neural network architectures. They also find that mixup reduces the memorization of corrupt labels, increases the robustness to adversarial examples, and stabilizes the training of generative adversarial networks.

3. Background study

Following what we have seen from the specialized literatures on learning from imbalanced data, two main methods are relevant for approaching imbalanced learning problem: Synthetic Minority Oversampling Technique ($SMOTE$) and mixup vicinal data augmentation. So the next two sections focus on those methods.

3.1 SMOTE: Synthetic Oversampling Methods

$SMOTE$ stands for *Synthetic Minority Oversampling Technique* [4] [5] and is a one of the data level methods, i.e it consists to balance the data by adding instances of the data to the minority classes. It is proposed to deal with the limit of random oversampling [5] which is related to the fact that the simple random oversampling can increase the likelihood of overfitting because it makes exact copies of existing instances [6].

In brief, its main idea is to create new minority class examples by interpolating several minority class instances that lie together for oversampling the training set. By doing so, take the difference between the feature vector, i.e sample, under consideration and its nearest neighbour. Then multiply this difference by a random number between 0 and 1 and add it to the feature vector under consideration. This is equivalent to introducing synthetic examples along the line segments joining any of the k minority class nearest neighbors. Depending upon the amount of over-sampling required, neighbors from the k nearest neighbors are randomly chosen. This process is illustrated in Figure 3.1 where x_i is the selected point, x_{i_1} to x_{i_4} are some selected nearest neighbors and r_1 to r_4 the synthetic data points created by the randomized interpolation [4] [5].

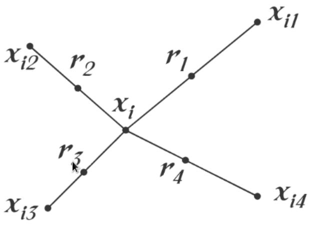


Figure 3.1: An illustration of how to create the synthetic data points in the SMOTE algorithm [4]

The advantages of this technique relies on its simplicity. $SMOTE$ is easy to understand and use: it is already implemented under the python toolbox *imbalanced-learn* [7]. Furthermore, on the contrary of random oversampling technique, $SMOTE$ alleviates some of overfitting problem because of the fact that it does not just copy the existing instances. However the weakness of $SMOTE$ is that it randomly chooses and generates the same number of synthetic data samples for each original minority example without considering the neighboring examples which increases the occurrence of overlapping between classes [8]. For that reason, various adaptive sampling methods have been proposed to overcome this limitation such as Borderline-$SMOTE$ [9], and Adaptive Synthetic Sampling ($ADASYN$)[10] algorithms.

3.2 Data augmentation: mixup method

3.2.1 Empirical Risk Minimization. Consdider a set of function $\mathcal{F}$. Consider X and Y a set of random input vectors and a set of target vectors respectively where the elements of X follows a fixed and unknown distribution P.

Let $x \in X$ and $y \in Y$. In supervised learning, the problem is to find a function $f \in \mathcal{F}$ that approximates the best the target response y. To achieve that, let's define a loss function ℓ and measure ℓ between the predictions $f(x)$ to actual target y. Consider the expected value of ℓ called *expected risk* [11] [12]:

$$R(f) = \int \ell(f(x), y) dP(x, y), \tag{3.2.1}$$

where $P(x, y) = P(y|x)P(x)$ is the joint probability of $(x, y) \in X \times Y$.

The goal is to minimize the risk functional $R(f)$ over the data distribution P. But unfortunately, since $P(x, y)$ is usually unknown, it is difficult to find a function to minimize the equation (3.2.1) directly. Instead, the only available solution for this problem is to approximate P by *empirical disribution* which is defined by the information contained in a set of training data $(x_i, y_i)_{i=1}^n$, where $(x_i, y_i) \sim P$ for all $i = 1, \ldots, n$:

$$P_{emp}(x, y) = \frac{1}{n} \sum_{i=1}^{n} \delta(x = x_i, y = y_i), \tag{3.2.2}$$

where δ is a Dirac mass centered at (x_i, y_i).

Now, we have the *empirical risk* R_{emp}:

$$R_{emp}(f) = \int \ell(f(x), y) dP_{emp}(x, y). \tag{3.2.3}$$

From the equation (3.2.2) to the equation (3.2.3),

$$\begin{aligned}
R_{emp}(f) &= \int \ell(f(x), y) dP_{emp}(x, y) \\
&= \int \ell(f(x), y) d(\frac{1}{n} \sum_{i=1}^{n} \delta(x = x_i, y = y_i)) \\
&= \frac{1}{n} \int \ell(f(x), y) \sum_{i=1}^{n} d(\delta(x = x_i, y = y_i)) \quad (derivation\ of\ sum's\ property) \\
&= \frac{1}{n} \sum_{i=1}^{n} \int \ell(f(x), y) d(\delta(x = x_i, y = y_i) \quad (finite\ sum\ property) \\
&= \frac{1}{n} \sum_{i=1}^{n} \int d\ell(f(x_i), y_i),
\end{aligned}$$

finally

$$R_{emp}(f) = \frac{1}{n}\sum_{i=1}^{n}\ell(f(x_i), y_i). \tag{3.2.4}$$

The Empirical Risk Minimization (ERM) is then equivalent to learn the function f by minimizing equation (3.2.4)as following:

$$min_{f\in\mathcal{F}}\ R_{emp}(f). \tag{3.2.5}$$

Inspite of its big success in many deep learning tasks, ERM technique has a major drawback: instead of generalizing the training data, it memorizes everything even the corrupt labels. This fact leads into a poor performance of the algorithm when we just use a new data outside the training set [11].

3.2.2 Vicinal Risk Minimization and Mixup.

Vicinal Risk Minimization

The empirical risk defined in the equation (3.2.3) can be equivalently rewritten as [14]

$$R_{vic}(f) = \int \ell(f(x), y)dP_{est}(x, y), \tag{3.2.6}$$

with the estimate distribution

$$P_{est}(x, y) = \frac{1}{n}\sum_{i=1}^{n}P_{x_i}(x).\delta_{y_i}(y), \tag{3.2.7}$$

where $\delta_a(x)$ stands for the Dirac mass centered at the point a. Since the support of $P_{x_i}(x)$ is an one-point set x_i $(i = 1, 2, \ldots, n)$, $P_{est}(x, y)$ cannot approximate $P(x, y)$ accurately if $P(x, y)$ is continuous. Thus, *the Vicinal Risk Minimization*(VRM) was proposed to improve the feasibility of ERM by replacing the Dirac mass with some vicinity function defined on the set $\Omega(x_i) = x : d(x, x_i) \leq r_{x_i}$ with respect to the metric $d(.,.)$, where the $\Omega(x_i)$ is called the vicinity of the point x_i [13] [14] .

In general, the equation (3.2.7) can be written as [11]

$$P_{est}(\tilde{x}, \tilde{y}) = \frac{1}{n}\sum_{i=1}^{n}\nu(\tilde{x}, \tilde{y}|x_i, y_i), \tag{3.2.8}$$

where ν is a vicinity distribution that measures the probability of finding the virtual feature-target pair $(\tilde{x}, \tilde{y})$ in the vicinity of the training feature-target pair (x_i, y_i).

Equation (3.2.8) to equation (3.2.6), we have *the empirical vicinal risk*:

$$\begin{aligned}
R_{vic}(f) &= \int \ell(f(x), y)d(\frac{1}{n}\sum_{i=1}^{n}\nu(\tilde{x}, \tilde{y}|x_i, y_i)) \\
&= \frac{1}{n}\int \ell(f(x), y)\sum_{i=1}^{n}d(\nu(\tilde{x}, \tilde{y}|x_i, y_i)) \quad (derivation\ of\ sum's\ property) \\
&= \frac{1}{n}\sum_{i=1}^{n}\int \ell(f(x), y)d(\nu(\tilde{x}, \tilde{y}|x_i, y_i)) \quad (finite\ sum\ property) \\
&= \frac{1}{n}\sum_{i=1}^{n}\int d\ell(\tilde{x}_i, \tilde{y}_i),
\end{aligned}$$

finally,

$$R_{vic}(f) = \frac{1}{n}\sum_{i=1}^{n} \ell(f(\tilde{x_i}), \tilde{y_i}). \tag{3.2.9}$$

The Vicinal Risk Minimization (VRM) principle is consists of estimating

$$min_{f \in \mathcal{F}} R_{vic}(f). \tag{3.2.10}$$

This technique improves generalization of the training data. However it requires big understanding of the distribution of the dataset in order to separate between classes before one can apply it [11].

Mixup

As the vicinal distribution ν can be chosen arbitrary, one of the recent study proposed is called *mixup vicinal distribution* [11] which is defined as follows:

$$\nu(\tilde{x}, \tilde{y}|x_i, y_i) = \frac{1}{n}\sum_{j}^{n} \mathbb{E}_{\lambda}[\delta(\tilde{x} = \lambda.x_i + (1-\lambda).x_j, \tilde{y} = \lambda.y_i + (1-\lambda).y_j)], \tag{3.2.11}$$

where $\lambda \sim Beta(\alpha, \alpha)$, for $\alpha \in (0, \infty)$.

Using this new distribution, the efficiency of the machine learning algorithm will be improved in term of adversarial examples [11].

4. Experiment

Now, we have seen some background knowledge about the methods to approach the imbalanced learning problem in order to increase the performance of our machine learning algorithm toward the imbalanced dataset. Accordingly, in this chapter, we will bring that knowledge into practice. Additionaly, we use binary classification only for this work for the reason of its simplicity.

To do that, this first section gives the descripton of the dataset that we use for this experimentation. In this work we use the credit card fraud dataset [15] which is taken from *kaggle* website [16].

4.1 Dataset

This dataset contains the European credit card transactions that occurred over two days. It has $284,807$ rows, with $284,315$ normal and 492 fraudulant transactions, and 31 columns. The choice of this dataset is to highlight the highly unbalance dataset as shown in figure 4.1.

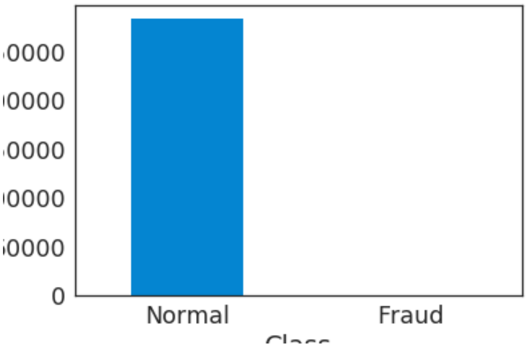
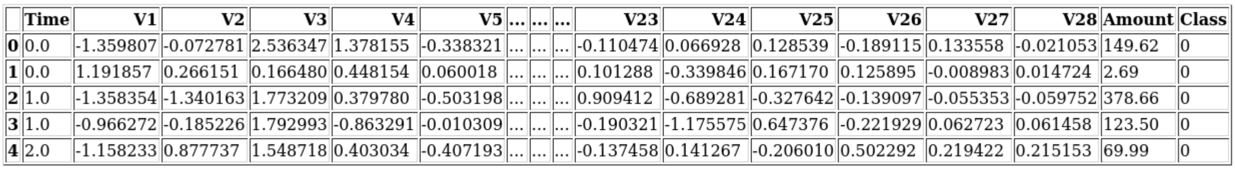


Figure 4.1: Very imbalanced dataset

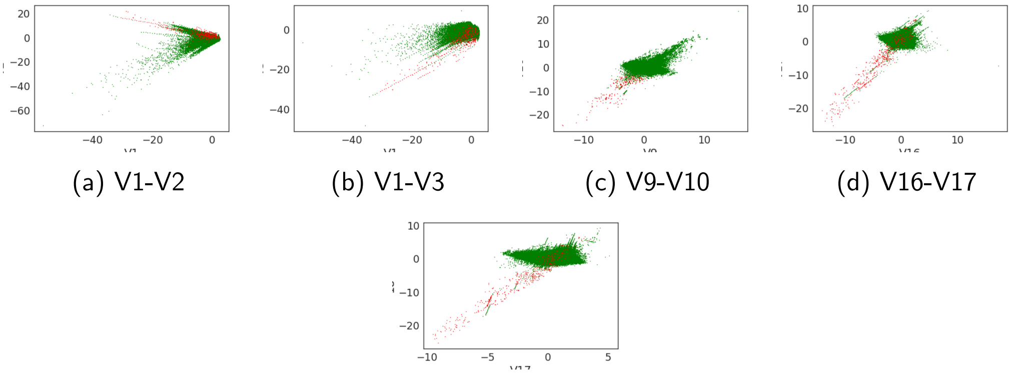
The 31 columns are equivalent to the input variables $V1, V2, \ldots, V28$, $Time$, $Amount$, and $Class$ which are numerical variables in which $Time$ is the seconds elapsed between each transaction and the first transaction in the dataset, $Amount$ is the transaction amount, and $Class$ is the response variable which takes value 1 in case of fraud and 0 otherwise [15] [16]. Figure 4.2 shows the overview of the data frame.



	Time	V1	V2	V3	V4	V5	...	...	...	V23	V24	V25	V26	V27	V28	Amount	Class
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	...	...	...	-0.110474	0.066928	0.128539	-0.189115	0.133558	-0.021053	149.62	0
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	...	...	...	0.101288	-0.339846	0.167170	0.125895	-0.008983	0.014724	2.69	0
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	...	...	...	0.909412	-0.689281	-0.327642	-0.139097	-0.055353	-0.059752	378.66	0
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	...	...	...	-0.190321	-1.175575	0.647376	-0.221929	0.062723	0.061458	123.50	0
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	...	...	...	-0.137458	0.141267	-0.206010	0.502292	0.219422	0.215153	69.99	0

Figure 4.2: Overview of the dataframe

The following plot (figure 4.3) shows the spread of fraud vs non-fraud on selected variables which has good spread of frauds and non frauds.



(a) V1-V2 (b) V1-V3 (c) V9-V10 (d) V16-V17

(e) V17-V18

Figure 4.3: The spread of fraud vs non-fraud on selected variables

So far, we have confirmed that our data is highly imbalanced. Now, let us split the dataset into 75% for training data in order to train our model and 25% testing data used to test the accuracy of the model.

4.2 Methods

4.2.1 Data samplers. This work is a comparative study of the two methods that we have discussed in chapter 3 and their extensions, with their hyperparameters, listed as the following:

- SMOTE
  - random_state=42,
  - ratio=1:25000,
- mixup
  - alpha = 0.1
- mixup + SMOTE,
  - alpha = 0.1,
  - random_state=42,
  - ratio=1:25000.

4.2.2 Classifiers. To evaluate the data samplers above, two classifiers are chosen to make sure that it works for the general classifiers but not only for specific classifier. Thus, as the choice is arbitrary, we pick *Random Forest* [17] and *Gradient Boost* [18] classifiers. This choice is relies on the fact that those algorithms reduce overfitting [17][18] so that we can focus only on the imbalanced learning. Equally important is their simplicity to use because they are already implemented in *scikit-learn* [19].

In addition, the hyperparameters of the classifiers must be fixed as we set in the following list:

- Random Forest

  - we use defaults

- Gradient Boost
  - params = {'n_estimators': 500, 'max_depth': 3, 'subsample': 0.5, 'learning_rate': 0.01, 'min_samples_leaf': 1, 'random_state': 3}

4.3 Evaluation Metrics

4.3.1 Confusion matrix. Confusion matrix is a table (shown in table 4.1) that is used for describing the performance of a classification model.

	True	False
True	TP	TN
False	FP	FN

Table 4.1: Confusion matrix

where:

- TP (True Positives): True labels which are predicted as True,
- TN (True Negatives): False labels which are predicted as False,
- FP (True Positives): False labels which are predicted as True, called also *type I error*.
- FN (True Positives): True labels which are predicted as False, called also *type II error*.

4.3.2 Accuracy. Accuracy is defined as the percentage of correct predictions [21] which is given by the following formula:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{4.3.1}$$

Considering that accuracy itself is not sufficient for evaluating the performance of imbalanced learning algorithm [21], we instead apply a set of assessment metrics such as precision, recall, specificity and $f1 - score$.

4.3.3 Classification report: precision, recall, $f1$-score.

Precision

Precision is the fraction of positive predictions that are correct [21] which is given by the following ratio:

$$Precision = \frac{TP}{TP + FP} \tag{4.3.2}$$

where TP represents True Positive and FP represents False Positive.

Recall

Recall is the fraction of the truly positive instances that the classifier recognizes, also called sensitivity, [21] which is calculated with the following ratio:

$$Recall = \frac{TP}{TP + FN} \tag{4.3.3}$$

where FN represents False Negative.

$f1$-score

$f1 - score$ measure is the harmonic mean, or weighted average, of the precision and recall scores [21] which is calculated using the following formula:

$$f1 - score = \frac{(1 + \beta^2) * Precision * Recall}{\beta^2 * Recall + Precision} \tag{4.3.4}$$

where β is a weight coefficient to adjust the significance of recall (usually $\beta = 1$).

4.3.4 Receiver Operating Characteristic and Area Under the ROC Curve.

Receiver Operating Characteristic (ROC)

ROC curve visualizes a classifier's performance. Unlike accuracy, the ROC curve is insensitive to data sets with unbalanced class proportions; unlike precision and recall, the ROC curve illustrates the classifier's performance for all values of the discrimination threshold. ROC curves plot the classifier's recall against its fall-out. Fall-out, or the false positive rate, is the number of false positives divided by the total number of negatives [21]. It is calculated using the following formula:

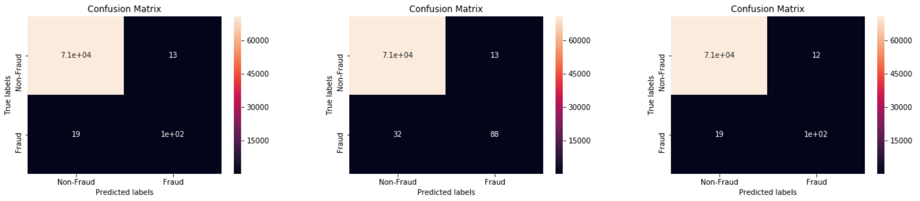
$$F = \frac{FP}{TN + FP} \tag{4.3.5}$$

Area Under the ROC Curve (AUC)

AUC is the area under the ROC curve; it reduces the ROC curve to a single value, which represents the expected performance of the classifier [21]. The highest the value of AUC the best is the model.

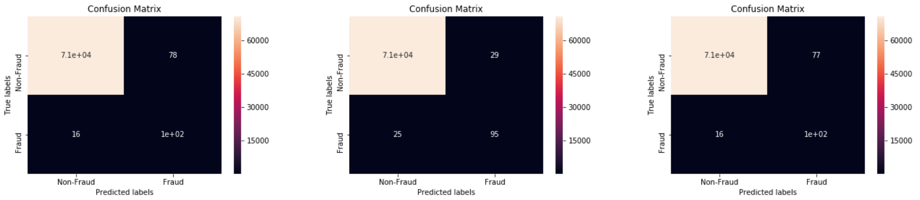
5. Experimental Results

Figure 5.1 and Figure 5.2 represent the confusion matrix of Random Forest and Gradient Boost classifiers respectively for the three methods. In fact, it is clear that the majority class has always the highest predicted score for every classifier. However we can see that even if the ratio of the data is very skewed, classifiers are able to predict the minority group normaly for every method.



(a) $SMOTE$ (b) mixup (c) mixup + SMOTE

Figure 5.1: CM of RF for the three methods



(a) SMOTE (b) mixup (c) mixup + $SMOTE$

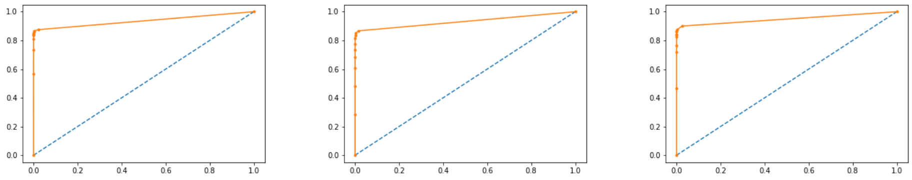
Figure 5.2: CM of GB for the three methods

To verify what we have seen from those confusion matrices, we need to check the accuracy of the classifiers. Computed from the equation (4.3.1), the table 5.1 shows the accuracy of Random Forest and Gradient Boost classifiers by using the three methods . It is then clear that both $SMOTE$ and mixup method perform well with our both classifiers in terms of accuracy. And when we look at the combination of the two methods, it is still having the high accuracy in the same way each of them does.

	$SMOTE$	mixup	mixup + $SMOTE$
Random Forest	0.999508	0.999157	0.999438
Gradient Boost	0.9986798	0.999073	0.996854

Table 5.1: Accuracy of Random Forest and Gradient Boost classifiers by using the three methods

In spite of those good accuracies that we have got, it is always better to have a look at another metrics to confirm the performance of the models. Figure 5.3 and Figure 5.4 give the AUC of each methods on RFC and GB classifiers. In general, as we can observe here, the AUC have all a high values which indicate the consistency with accuracies especially from using the combination of $SMOTE$ and mixup.

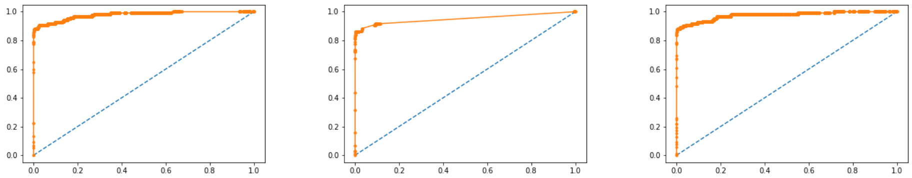


(a) $SMOTE$

(b) mixup

(c) mixup + $SMOTE$

Figure 5.3: AUC of RF for the three methods



(a) $SMOTE$

(b) $mixup$

(c) mixup + $SMOTE$

Figure 5.4: AUC of GB for the three methods

6. Conclusion

Imbalanced dataset is one of the big issues in machine learning. Balancing the distribution of the training data seems to be the best way to approch the imbalanced learning problems. Such techniques are known as *sampling* techniques which have two categories: undersampling and oversampling. $SMOTE$ is one of the best among them and it can be confirmed from the accuracy performed with $SMOTE$. However when we look closer at the precision, it seems $SMOTE$ lost a bit of performance. It can be seen from the comparison between $SMOTE$ and mixup results.

The proposed method (mixup + $SMOTE$) exhibits high values of AUC but still having almost the same accuracy with each method. So, in one word, combining both methods seems to be a prominent in the field of imbalanced learning.

Further work may consequently focus on applying mixup + $SMOTE$ to multiclass problems. Additionaly, tunning the best values of every hyperparameters used, like α and ratio used for resampling the data with $SMOTE$, should be done from further analysis.

Bibliography

[1] Simple English Wikipedia: *Machine Learning*, $https : //simple.wikipedia.org/wiki/Machine_{l}earning$

[2] Haibo He (Member IEEE), and Edwardo A. Garciá, 2009. *Learning from Imbalanced Data.*

[3] Bartosz Krawczyk. Learning from imbalanced data: open challenges and future directions. ISSN 2192-6360, Doi10.1007/s13748-016-0094-0.

[4] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, W. Philip Kegelmeyer, 2002. *SMOTE: Synthetic Minority Over-sampling Technique.*

[5] Alberto Fernández, Salvador Garciá, Francisco Herrera, Nitesh V. Chawla, 2018. *SMOTE for Learning from Imbalanced Data: Progress and Challenges, Marking the 15-year Anniversary.*

[6] Mr.Rushi Longadge, Ms. Snehlata S. Dongre, Dr. Latesh Malik, 2013. *Class Imbalance Problem in Data Mining: Review.*

[7] Guillaume Lemaître, Fernando Nogueira, Christos K. Aridas, 2017. *Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning*.

[8] B.X. Wang, N. Japkowicz, Imbalanced data set learning with synthetic samples, in: Proceedings of the IRIS Machine Learning Workshop, 2004.

[9] H. Han, W.Y. Wang, B.H. Mao, Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning, in: Proceedings of the 2005 International Conference on Intelligent Computing (ICIC'05), Lecture Notes in Computer Science, vol. 3644, 2005, pp. 878-887.

[10] H. He, Y. Bai, E.A. Garcia, S. Li, ADASYN: adaptive synthetic sampling approach for imbalanced learning, in: Proceedings of the 2008 IEEE International Joint Conference on Neural Networks (IJCNN'08), 2008, pp. 1322-1328.

[11] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, David Lopez-Paz. 2018. *mixup: BEYOND EMPIRICAL RISK MINIMIZATION.*

[12] V. Vapnik, *Principles of Risk Minimization for Learning Theory.*

[13] Min-Hsiu.Hsieh , Dacheng Tao , Chao Zhang, *Generalization Bounds for Vicinal Risk Minimization Principle*

[14] Olivier Chapelle, Jason Weston, Léon Bottou and Vladimir Vapnik, *Vicinal Risk Minimization*, AT&T research Labs, 100 Schultz drive, Red Bank, NJ, USA.

[15] Andrea Dal Pozzolo, Olivier Caelen, Reid A. Johnson and Gianluca Bontempi. Calibrating Probability with Undersampling for Unbalanced Classification. In Symposium on Computational Intelligence and Data Mining (CIDM), IEEE, 2015.

[16] Kaggle Datasets, https://www.kaggle.com/datasets

[17] Gérard Biau, Erwan Scornet, *A Random Forest Guided Tour*, arXiv:1511.05741 [math.ST]

[18] Janek Thomas, Stefan Coors, Bernd Bischl, *Automatic Gradient Boosting*, Department of Statistics, LMU, Ludwigstrasse 33, D80539 Munich.

[19] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot and Edouard Duchesnay, *Scikit-learn: Machine Learning in Python*, Journal of Machine Learning Research 12 (2011) 2825-2830.

[20] Ayyadevara V.K. (2018) Gradient Boosting Machine. In: Pro Machine Learning Algorithms. Apress, Berkeley, CA.

[21] N. Japkowicz, *"Assessment metrics for imbalanced learning," Imbalanced learning: Foundations, algorithms, and applications*, pp. 187-206, 2013.