Models of classification in context of imbalanced datas

Thibault Fuchez

2021-07-31

Contents

Chapter 1

Introduction

1.1 Abstract

In data-minig, a frequent and relevant issue to handle with two-class classification is to make reliable predictions with strongly imbalanced distribution of the target variable. Most of the machine learning algorithms assume by default that all data are balanced. Empirically, a lot of datasets faces this distortion (fraud detection, anticipation of catastrophes, donators in case of funding campain, unusual returns on stock markets, ...). The majority class represents "normal cases", while the minority class represents "abnormal" cases. Because The least common values of the target variable are linked with events which are very relevant for users, we considered the minority class as positive and the majority class as negative.

The commun issue with classifiers is that they are unable to learn from the positive class. It results that the predictions are almost only negative class. Algorithms failed to predict the positive class which is properly what the users need.

Nowadays, the imbalanced data sets problem plays a key role in machine learning. During the last decades, literature was very prolific about this subject. Many tools were developped to solve this problem. This paper has neither ambition to give an exhaustive review of the existing solutions nor exploring new solutions. Moreover, we won't go to far in the explanation of the mathematical principle handling the algorithms. Our purpose is to propose some elements of solution to counteract the effect of imbalanced datatset which can be used an uderstanded by people who like data-minig without having a large scale of knowledge in this domain.

1.2 Document format

Chapter 1 was an introduction of this paper.

On chapter 2, we introduce some tools which allows to measure the efficiency of a model.

Chapter 3 briefly presents the different models we used. We will touch upon the math behind the classifier and the way the algorithm works.

Chapter 4 introduces some remedies to make our classifiers better predictors.

Chapter 5 is an application on three dataset with which we can evaluate the submitted remedies.

Chapter 6 stands as a conclusion.

Chapter 2

Metrics for classification tasks

Understanding if a model is a good classifier or not requires a good comprehension of the predictions. We want to know the global effiency but some specific element too. Is classifier better in predicting one or the other class? What are the strength and weakness of the model? Can we be confident towards the results, haw can we know they are not due to chance?

In order to compare the performance's models, we use different "metrics". Our tools reliability is highly dependent on the data structure. In our case, the imbalanced sample of data classes has to be taken into account in order to find the metrics which allows to give a good evaluation of our models.

2.1 The foundation of the metrics: The Confusion Matrix

The confusion matrix presents the results obtained by a given classifier. This table provides the instances that were correctly classified (True Positive and True negative), and the instances that were wrongly classified (False Positive and False negative). From this table, we can calculate all the metrics described below

	Predicted	
	Positive	Negative
True		
Positive	TP	FN
Negative	FP	TN

2.2 Accuracy and error rate

$$error = \frac{FP + FN}{TN + TP + FP + FN}$$

$$accuracy = 1 - error$$

The first metrics is obviously the global accuracy and its complement the error rate. It is the most frequently used to estimate the performance of a model. If accuracy is too low, we deduce that our learning algorithm is globally inefficient. However In the context of imbalanced data set, accuracy is not suitable. Indeed, because of the massive representation of the negative class, and as the classifiers failed to identify the positive class, we reach a high value of accuracy. For instance, if only 10% of the cases belong to the positive class and the classifiers predicts all cases as negative, accuracy will be at 90%. This is worthless when users objectives is to predict the rare cases.

To reflect more closely the users needs and priorities, several performance measure exist.

2.3 True Positive rate

$$TP_{rate} = \frac{TP}{TP + FN} = \frac{TP}{P_{real}}$$

Also called sensitivity, recall or detection power. I personally prefer the term detection power because it is more explicit. It is the ratio of the value predicted as positive and which are actually positive among all the real positive. This is the ability of our classifier to detect the positive cases.

2.4 True Negative and False positive rate

Also called specificity. It is the ratio of the value predicted as negative and which are actually negative among all the real negative case.

$$TN_{rate} = \frac{TN}{TN + FP}$$

I prefer its complement, the False positive rate, also called False alarm. Indeed, the term "False alarm" is more relevant than specificity.

$$FP_{rate} = \frac{FP}{TN + FP} = 1 - TN_{rate}$$

TPrate (detection power) and FP rate (False alarm) are often quote in the literature as benefits and costs, respectively. These terms refers to a central

point of our problematic. Indeed, a key point to find good remedies is to make a trade-off between what it cost in terms of False alarm and the benefits gained in terms of detection power.

2.5 Positive prediction value : Precision

$$PP_{value} = \frac{TP}{TP + FP} = \frac{TP}{P_{pred}}$$

The precision measures the rate of True positive among all cases predicted as positive.

2.6 F-measure

The F-measure is a combination of both precision and recall. This metric value is high when both recall (a measure of completeness) and precision (a measure of exactness) are high. Hence, this metrics is particulary suitable on predicting the case that matter to the user.

$$F_\beta = \frac{(1 + \beta^2) \times recall \times precision}{\beta^2 \times recall + precision}$$

Beta is a coefficient to adjust the weight of recall against precision. In this paper, we choose a value of 1 which give the same weights to recall and precision.

2.7 Kappa

$$K = \frac{P_{agree} - P_{chance}}{1 - P_{chance}}$$

$$P_{agree} = \frac{TP + FN}{number\ of\ cases}$$

$$P_{chance} = \frac{P_{pre} \times P_{act}}{number\ of\ cases^2} + \frac{N_{pre} \times N_{act}}{number\ of\ cases^2}$$

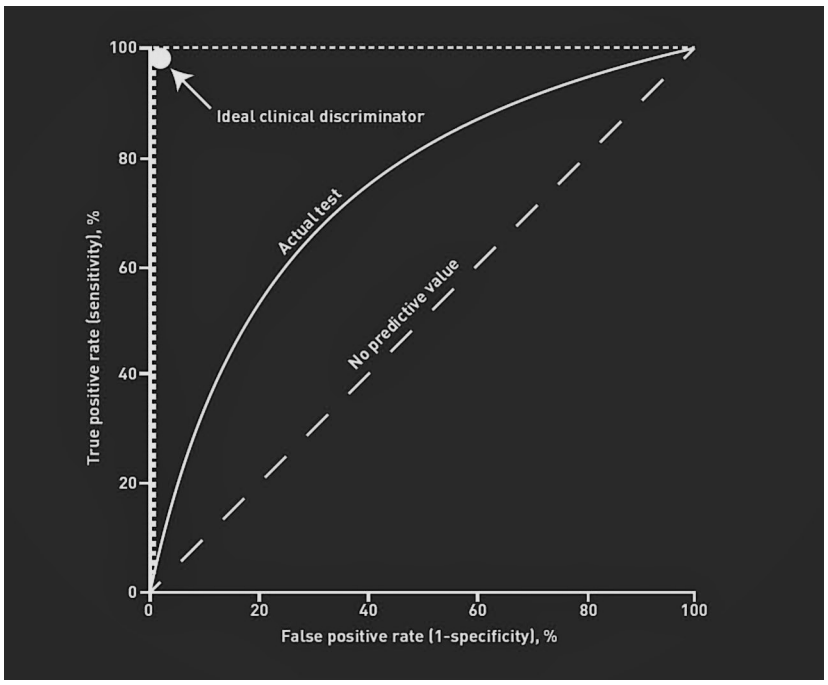
Kappa is a very interesting metrics in context of imbalanced datas. The calculation is based on the difference between how much agreement(positive) is actually present ("observed") compared to how much positive would be expected to be present by chance alone ("expected"). We want to know how different the observed positive are from the expected. Kappa is a measure of this difference(McHugh, 2015).

score	app
< 0	Less than chance agreement
0.01-0.20	slight agreement
0.21– 0.40	Fair agreement
0.41–0.60	Moderate agreement
0.61–0.80	Substantial agreement
0.81–0.99	Almost perfect agreement

2.8 ROC Curve

A receiver operating characteristic curve, or ROC curve, is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied.

The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings.



Chapter 3

Classifiers

3.1 LDA

3.2 Presentation

The Linear Discriminant analysis classifier is used to predict the probability of belonging to a given class based on one or multiple predictor variables. It works with continuous and/or categorical predictor variables.

Linear discriminant analysis is a generalization of Fisher's linear discriminant, a method used in statistics to find a linear combination of features that characterizes or separates two or more classes of objects.

In this approach, Fishers sought to find the linear combination of the predictors such that the between-group variance was maximized relative to the within-group variance. In other words, he wanted to find the combination of the predictors that gave maximum separation between the centers of the data while at the same time minimizing the variation within each group of data. (Li, 2014)

3.2.1 Learning LDA model

μ_k and σ_k the mean and variance for the k-class. LDA makes predictions by estimating the probability that a new set of inputs belongs to each class. LDA makes some simplifiying assumptions about your data: * that your data are gaussian * that each distribute the same variance, so $\sigma_0 = \sigma_1 = \sigma$ The model uses Bayes Theorem to estimate the probabilities.

$$P(Y = c|X = x) = \frac{\pi_c f_c(x)}{\sum_{j=1}^{k} \pi_j f_j(x)}$$

Where π_c refers to the base probability of each class (c) observed in your training data. In Bayes' Theorem this is called the prior probability.

$$\pi_c = \frac{n_c}{n}$$

Estimation of f_c is more delicate, it is the estimated probability of x belonging to the class. A Gaussian distribution function is used for f_c. Plugging the Gaussian into the above equation and simplifying we end up with the equation below. This is called a discriminate function and the class is calculated as having the largest value will be the output classification :

x will be affected to the class c for which:

$$\hat{\delta}_c(x) = \ln \hat{\pi}_c - \frac{\hat{\mu}_c^2}{2\hat{\sigma}^2} + x\frac{\hat{\mu}_c}{\hat{\sigma}^2}$$

is maximum.

$\hat{\delta}_c(x)$ is the discriminate function for class c given input x, the mu_k, $sigma^2$ and π_c are all estimated from your data.

3.3 logistic regression

In statistics, the logistic model (or logit model) is used to model the probability of a certain class. Each object being detected in the image would be assigned a probability between 0 and 1, with a sum of one. Logistic regression is a statistical model that in uses a logistic function to model a binary dependent variable. In regression analysis, logistic regression is estimating the parameters of a logistic model.

Binary Logistic Regression Major Assumptions: * The dependent variable should be dichotomous in nature. * There should be no outliers in the data, which can be assessed by converting the continuous predictors to standardized scores * There should be no high correlations (multicollinearity) among the predictors. This can be assessed by a correlation matrix among the predictors. Tabachnick and Fidell suggest that as long correlation coefficients among independent variables are less than 0.90 the assumption is met (Tabachnick and Fidell, 2007).

At the center of the logistic regression analysis is the task estimating the log odds of an event. Mathematically, logistic regression estimates a multiple linear regression function defined as:

$$logit(p) = \log \frac{p(y=1)}{1-p(y=1)} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_i x_i$$

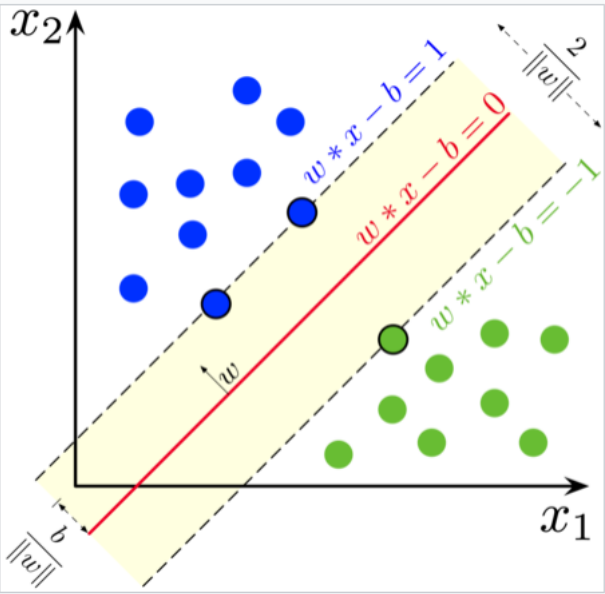
Overfitting. When selecting the model for the logistic regression analysis, another important consideration is the model fit. Adding independent variables

to a logistic regression model will always increase the amount of variance explained in the log odds (typically expressed as R^2). However, adding more and more variables to the model can result in overfitting, which reduces the generalizability of the model beyond the data on which the model is fit.

3.4 SVM

Support vector machine try to make a reasonable choice as the best hyperplane is the one that represents the largest separation, or margin, between the two classes. So we choose the hyperplane so that the distance from it to the nearest data point on each side is maximized. If such a hyperplane exists, it is known as the maximum-margin hyperplane and the linear classifier it defines is known as a maximum-margin classifier; or equivalently, the perceptron of optimal stability.More formally, a support-vector machine constructs a hyperplane or set of hyperplanes in a high dimensional space, which can be used for classification, regression, or other tasks like outliers detection. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general the larger the margin, the lower the generalization error of the classifier.

The support vector machines create an optimum hyperplane that separates the training data by the maximum margin. However, sometimes we would like to allow some misclassifications while separating categories. The SVM model has a cost function, which controls training errors and margins. For example, a small cost creates a large margin (a soft margin) and allows more misclassifications. On the other hand, a large cost creates a narrow margin (a hard margin) and permits fewer misclassifications. In this recipe, we will illustrate how the large and small cost will affect the SVM classifier.



Maximum-margin hyperplane and margins for an SVM trained with samples from two classes. Samples on the margin are called the support vectors.

3.5 classification tree

Tree-based models consist of one or more nested if-then statements for the predictors that partition the data. Within these partitions, a model is used to predict the outcome.

Choosing the trees split points :

Technically, for regression modeling, the split cutoff is defined so that the residual sum of squared error (RSS) is minimized across the training samples that fall within the subpartition.

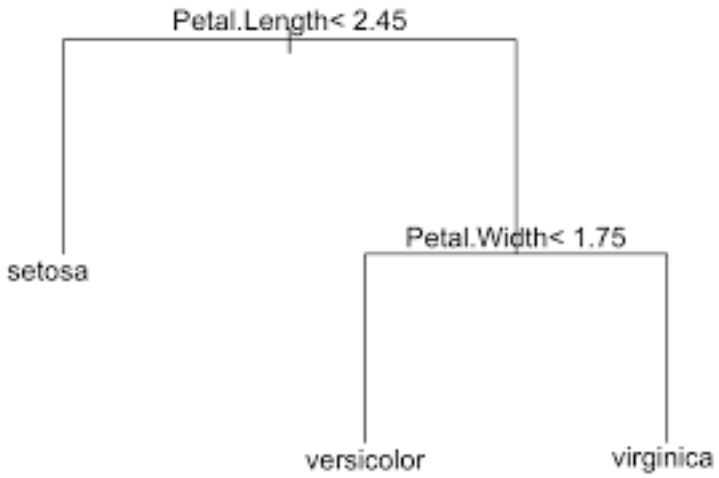
Recall that, the RSS is the sum of the squared difference between the observed outcome values and the predicted ones, $RSS = \sum((Observeds - Predicteds)^2)$.

In classification settings, the split point is defined so that the population in subpartitions are pure as much as possible. Two measures of purity are generally used, including the Gini index and the entropy (or information gain).

For a given subpartition, $Gini = \sum(p(1-p))$ and $entropy = -1 \times \sum(p * log(p))$, where p is the proportion of misclassified observations within the subpartition.

The sum is computed across the different categories or classes in the outcome variable. The Gini index and the entropy varie from 0 (greatest purity) to 1

(maximum degree of impurity)



3.6 random forest

RF classifier is an ensemble method that trains several decision trees in parallel with bootstrapping followed by aggregation, jointly referred as bagging. Bootstrapping indicates that several individual decision trees are trained in parallel on various subsets of the training dataset using different subsets of available features. Bootstrapping ensures that each individual decision tree in the random forest is unique, which reduces the overall variance of the RF classifier. For the final decision, RF classifier aggregates the decisions of individual trees; consequently, RF classifier exhibits good generalization.

3.7 Naives bayes

basics of formula :

$$P(A \cap B) = P(A)P(B|A) \iff P(B|A) = \frac{P(A \cap B)}{P(A)}$$

In statistics, naive Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naïve) independence assumptions between the features

Naives : The joint probability calculation is simpler for independent events. so we consider events are independent. (it will be too complexe for more than two events)

Laplace correction will allow a small chance for these types of unforeseen circumstances (if joint event probabilty equals to 0.)

Chapter 4

Remedies

A lot of research have been made concerning this problem. Our goal is not to make an exhaustive review of all the technics to remedies this issue. In this study, we will focus on methods than we can reproduce with our level of competence. It appears to us that it is interesting to separate the chosen methods in three levels :

- First, some remedies we can use before launching the machine learning algorithm (Pre-processing).
- Secondly, some remedies we can use during the computation of a fitted models by the machine (learning method tuning).
- At last, some remedies that can be used after the machine learning algorithm (post-processing).

4.1 Pre processing resampling

A lot of method exists in order to balanced a sample. Since the begin of the 21th century, a lot of studies appears in order to manage and discuss about sampling. Some methods are more sophisticated than other and might give better results, but they have no consensus on a method which have better results. Globally we can say it is the most common and popular remedies to counteract imbalanced data for two key reasons : we can use the sampled datas with all classifiers and it gives generally pretty goods results.

The first "basic" try of resampling is obviously try to make random over sampling or down-sizing. Over sampling consists in duplicate randomly positive cases in order to reach two class with approximately the same number of case. Down-sizing consists at randomly remove some negative cases in order to diminish the negative classes. Then we can easily imagine other way to select the cases removed or duplicate. Moreover, we can try to mix the two methods.

In her paper, Japkowicz discussed the effect of imbalance in a data set. Two resampling methods were considered. Random resampling consisted of resampling the smaller class at random until it consisted of as many samples as the majority class and "focused resampling" consisted of resampling only those minority examples that occurred on the boundary between the minority and majority classes. Random under-sampling was considered, which involved under-sampling the majority class samples at random until their numbers matched the number of minority class samples; focused under-sampling involved under-sampling the majority class samples lying further away. She noted that both the sampling approaches were effective, and she also observed that using the sophisticated sampling techniques did not give any clear advantage in the domain considered. (Japkowicz, 2000)

One approach that is particularly relevant to our work is that of Ling and Li (1998). They combined over-sampling of the minority class with under-sampling of the majorityclass. (Ling et al., 1998)

in order to try and discuss those resampling proceeds, the part "application" shows examples of random up and down sampling in order to make comparison with more sophisticated method as SMOTE and ROSE.

4.1.1 Random Up and Down sample :

DownSizing will randomly sample a data set so that all classes have the same frequency as the minority class. UpSample samples with replacement to make the class distributions equal.

4.1.2 ROSE

Creates a sample of synthetic data by enlarging the features space of minority and majority class examples. Operationally, the new examples are down from a conditionnal kernel density estimate of the two classes as describe in Menardi and torelli (2013).

The ROSE strategy to deal with class imbalance ROSE (Menardi and Torelli, 2014) provides a unified framework to deal simultaneously with the two abovementioned problems of model estimation and accuracy evaluation in imbalanced learning. It builds on the generation of new artificial examples from the classes, according to a smoothed bootstrap approach. Consider a training set T_n, of size n, whose generic row is the pair (x_i, y_i), i = 1, . . . , n. The class labels y_i belong to the set $(\gamma 0, \gamma 1)$, and x_i are some related attributes supposed to be realizations of a random vector x defined on R^d, with an unknown probability density function $f(x)$. Let the number of units in class $\gamma_j, j = 0, 1$, be denoted by $n_j < n$. The ROSE procedure for generating one new artificial example consists of the following steps:

- 1. Select $y* = Y_j$ with probability π_j.

- 2. Select $(xi, yi) T__n, *suchthat*y_i = y*$, with probability $\frac{1}{n_j}$.
- 3. Sample x^* from $K_{H_j}(., xi)$, with K_{H_j} a probability distribution centered at xi and covariance matrix Hj.

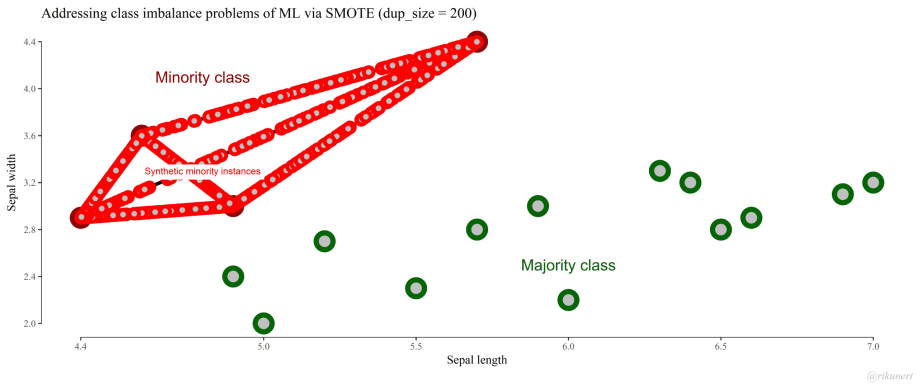
Essentially, we draw from the training set an observation belonging to one of the two classes, and generate a new example $(x*, y*)$ in its neighborhood, where the shape of the neighborhood is determined by the shape of the contour sets of K and its width is governed by Hj.

4.1.3 Smote family (smoteNC, BLSMOTE, ADASYN)

SMOTE is a well-known algorithm to fight this problem. The general idea of this method is to artificially generate new examples of the minority class using the nearest neighbors of these cases. Furthermore, the majority class examples are also under-sampled, leading to a more balanced dataset.

The SMOTE function oversamples your rare event by using bootstrapping and k-nearest neighbor to synthetically create additional observations of that event. The definition of rare event is usually attributed to any outcome variable that happens less than 15% of the time.

The minority class is over-sampled by taking each minority class sample and introducing synthetic examples along the line segments joining any/all of the k minority class nearest neighbors. Depending upon the amount of over-sampling required, neighbors from the k nearest neighbors are randomly chosen. The implementation currently uses five nearest neighbors. For instance, if the amount of over-sampling needed is 200%, only two neighbors from the five nearest neighbors are chosen and one sample is generated in the direction of each. Synthetic samples are generated in the following way: Take the difference between the feature vector (sample) under consideration and its nearest neighbor. Multiply this difference by a random number between 0 and 1, and add it to the feature vector under consideration. This causes the selection of a random point along the line segment between two specific features. This approach effectively forces the decision region of the minority class to become more general. (chawla, 2002)



	Predicted	
	Positive	Negative
True		
Positive	C(1, 1) TP	C(1,0) FN
Negative	C(0, 1) FP	C(0,0) TN

4.2 Learning method tuning

4.2.1 Metaparameters tuning

A first try can be to handle the parameters of the classifiers used. Empirycally we'll see in part 5 "Application" that it don't give some significant results to counteract imbalanced data set predictions.

4.2.2 Weighted class

We try to attribute some weights to our class but the results was not convinced. We try different ways and look for serious or significant results with this methods but it seems that it don't give results. Maybe the algorithm incorporing these results must be reshaped.

The best try to handle with classifiers during the learning step was to incorporate a cost matrix. We choose to use it with decision tree because the algorithm CART shows good results in the literrature.(Kuhn and Jonshon, 2013)

4.2.3 direct sensitive learning with cart classifer

The incorporation of benefits and/or costs (negative benefits) in existing algorithms, as a way to express the utility of different predictions, is one of the known approaches to cope with imbalanced domains.

Instead of optimizing the typical performance measure, such as accuracy or impurity, some models can alternatively optimize a cost or loss function that deferentially weights specific types of errors.

We summarize the theory of cost-sensitive learning, published mostly in (Elkan, 2001). The theory describes how the misclassification cost plays its essential role invarious cost-sensitive learning algorithms.

A cost matrix assigns a cost to each cell in the confusion matrix. The example below is a cost matrix where we use the notation C() to indicate the cost, the first value represented as the predicted class and the second value represents the actual class. (Elkan, 2002)

We can see that the cost of a False Positive is C(1,0) and the cost of a False Negative is C(0,1).

An intuition from this matrix is that the cost of misclassification is always higher

than correct classification, otherwise, cost can be minimized by predicting one class.

Conceptually, the cost of labeling an example incorrectly should always be greater than the cost of labeling it correctly.

$$TotalCost = C(0,1) * FalseNegatives + C(1,0) * FalsePositives$$

This is the value that we seek to minimize in cost-sensitive learning, at least conceptually.

To go further, a very good paper here. (Ling and Sheng, 2008)

4.3 post-processing threesholding

When there are two possible outcome categories, another method for increasing the prediction accuracy of the minority class samples is to determine alternative cutoffs for the predicted probabilities

The threshold probability (kind of classification criteria):

$$P(Y = 1, X = x) > 0,5$$

0.5 is a basic threshold which means that objects predicted with a probability to belong class 1 is higher than 0.5 will be classify as positive class. We can, for example, be more selective in terms of entry criteria and choose a threshold at 0,7. There may be situations where the sensitivity/specificity trade-off can be accomplished without severely compromising the accuracy of the majority class (which, of course, depends on the context of the problem).

Chapter 5

Applications

5.1 Introduction

In order to illustrate and discuss the different remedies proposed in the previous chapter, we are handling each on different dataset. Hence we can make comparisons and try to measure their efficiency.

- Few words about classifiers:

Our first choice as classifiers was to use LDA, LR, RF and SVM. having ascertained that LDA et LR give very similar results, we decide to substitute LR by naives bayes'classifier in order to proposed a richer experience. Notice that we first try to use glmnet instead of glm but it doesn't deliver better results (see spot.rmd). It is not unexpected that LR and LDA give nearly predictions, indeed they both are linear models and litteracy confirms they both give quite similar results.

- Few words About the code :

We don't introduce here all the manipulations done on the datasets, either the preparation of the dataset. You can find them in this github repositery wich contains the .rmd for each dataset. In this repositery, you can also find the .R file which contains also the functions we code in order to avoid to many repetition in the code. At last, the alldat.Rdata stocked all objects built in the .rmd, it is used here to call the object we need. A fourth data set is in development that is not use here (creditcard).

Our first think was to put the tuning of parameters inherent to classifiers as a track of remedies. We eventually decided to launch our first models with the parameters already tune. First because there was no significant improvement in detection power, then because svm models was unable to give results with the basic parameters.

Table 5.1: Table of priors ratio between positive and negative class

	spot	hack	recid
0	0.807562076749436	0.954514596062458	0.88374113218474
1	0.192437923250564	0.0454854039375424	0.11625886781526

Table 5.2: Confusion matrix spotify

	0	1	Sum	0	1	Sum
	rf			bayes		
0	4431	975	5406	4373	983	5356
1	147	115	262	205	107	312
Sum	4578	1090	5668	4578	1090	5668
	lda			svm		
0	4559	1078	5637	4566	1075	5641
1	19	12	31	12	15	27
Sum	4578	1090	5668	4578	1090	5668

To illustrate some remdies introduced above, We choose three dataset with different level of imabalanced.

Let's briefly presents those datasets:

- Spotify : a database including songs from spotify servers with categorical (about the qualification genre) and numerical predictors (about sound aspect of the songs computed by algorithm). Here the predicted variable is the popularity (is low /is not low).
- Recidivism : a database containing arrested people and some caterogical predictors about their identity and judicial priors.
- Hacked : a database concerning system and hacked tentative. a variable multiple_offense with "1" if hacked and "0" if not.

The table 5.1 shows the priors of the depending class for the three datasets.

5.2 First Models

We start to compute classifiers algorithm .For now, we just want to observe results with basic parameters. This first computation can be used as a start reference to measure the remedies tested later.

The table 5.2 shows the confusion matrix resulting to the four classifiers used on the spotofy dataset. We observe the unabilty to properly predict the unpopular songs. A look on the metrics (table5.3) sharps this observation.

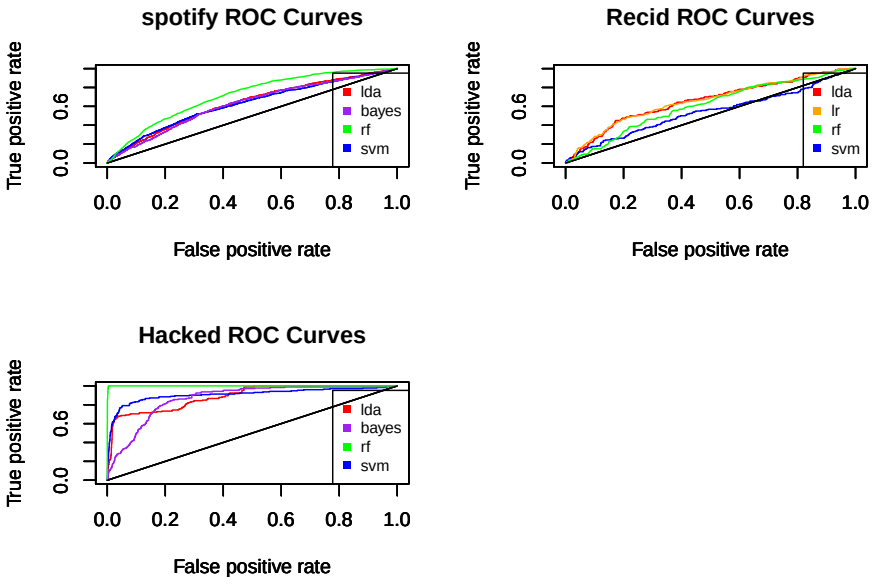
First we note that accuracy is very good, wich confirms accuracy is not a relaible metrics concerning imbalance dataset. A simple view on Detection power(TPr)

Table 5.3: Spotify metrics

	accuracy	FNrate	TPrate	kappa	PrecisionPPV	Fscore
rf	0.8020466	0.8944954	0.1055046	0.1032829	0.4389313	0.1701183
Bayes	0.7904023	0.9018349	0.09816514	0.07332293	0.3429487	0.1526391
lda	0.8064573	0.9889908	0.01100917	0.01088917	0.3870968	0.02140946
svm	0.8082216	0.9862385	0.01376147	0.01772557	0.5555556	0.02685765

shows that we don't achieve to predict what songs are very unpopular. FN rate is obviously good because of the imblalanced ratio.

Let see the plot curve for all datasets.



Spotify and recid faces a real problem, no classifiers is able to make good predictions. Hacked data set has pretty good results for a first try. It is possible that the nature of the data explains a part of this gap. Maybe variables can have some colinerarity or just don't have a real dependency for the dependant variable.

On table 5.4, we can see that only RF has very good results. Detection power and Fscore allows to judge the performance classifiers, contrary to accuracy or FN rate which are not usefull here. RF has the better results, In second position comes svm, even if he detects only a third among True positive, its quite a good results for a first try with an imbalanced data set, and Fscore ank kappa are not so bad.

Table 5.4: Hacked metrics

	accuracy	FNrate	TPrate	kappa	PrecisionPPV	Fscore
rf	0.9949815	0.05813953	0.9418605	0.9419781	0.9473684	0.9446064
Bayes	0.9477021	0.872093	0.127907	0.1597342	0.3142857	0.1818182
lda	0.9537771	0.7732558	0.2267442	0.287576	0.4814815	0.3083004
svm	0.9635499	0.7034884	0.2965116	0.4098061	0.75	0.425

Table 5.5: Spot with Smote-NC

	accuracy	FNrate	TPrate	kappa	PrecisionPPV	Fscore
rf	0.8278724	0.2075142	0.7924858	0.6557449	0.8528444	0.821558
Bayes	0.6241809	0.2647444	0.7352556	0.2483617	0.6016086	0.6617517
lda	0.6021188	0.3934032	0.6065968	0.2042377	0.6012124	0.6038926
svm	0.7035824	0.2505461	0.7494539	0.4071647	0.6864746	0.7165831

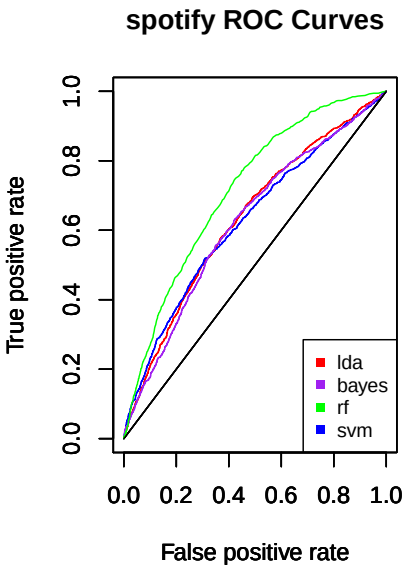
5.3 Preprocessing: Resampling methods

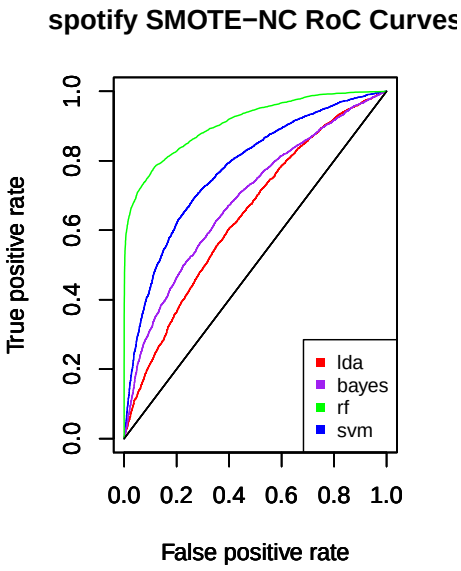
As we can see with the function *resamp* (in Funclib.R of thisgithub repositery)the funclin.R and the .rmd, we try many resampling with different method. As the smote methods seems to us pertinent with this approach of oversampling with some logic, we try some development as adasyn or BLSMOTE. We made some test with random up sampling and downsizing which don't stands here.

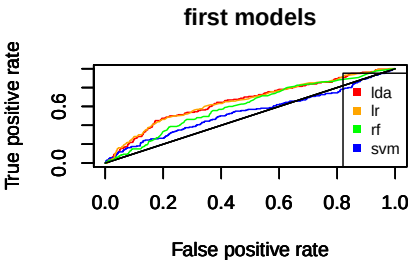
Note that the original smote algorithm is only for numerical variable. Some databases have few or no numerical variable so it is not an issue but a dataset as recid (with many caterogicla variable) asks to act on this variable. We could try to transform them as numerical but we choose to use SMoteNC, which is developed to deal with bot numerical and categorical variable.

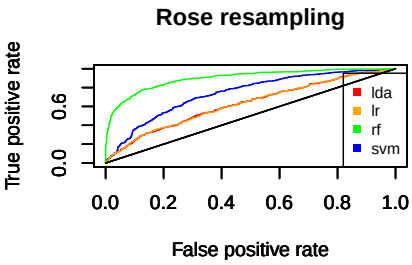
Table 5.5 shows the results of SMOTE-NC resampling.

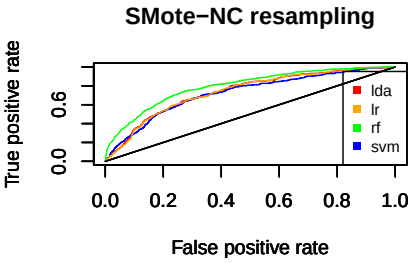
On graphics below, Lets compare RocCurve before and after smote resampling. This graphics illustrate a situation we faces a lot during our works. It often occurs with all smotefamily resamplers and Upsampling. We clearly see a global improvement. Random forest leads the course, following by SVM.


spotify ROC Curves
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0
lda
bayes
rf
svm


spotify SMOTE−NC RoC Curves
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0
lda
bayes
rf
svm


first models
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0
lda
lr
rf
svm


Rose resampling
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0
lda
lr
rf
svm

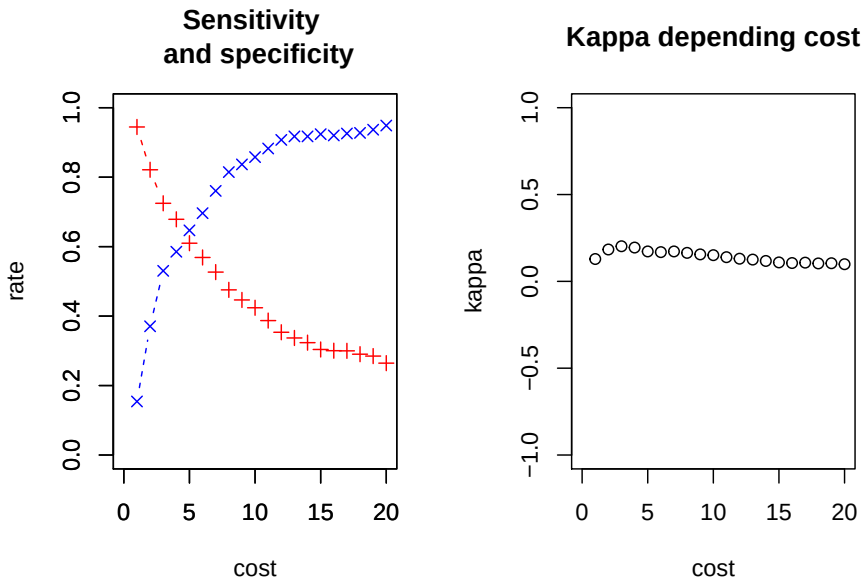

SMote−NC resampling
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0
lda
lr
rf
svm

5.4 Direct cost sensisitive learnig

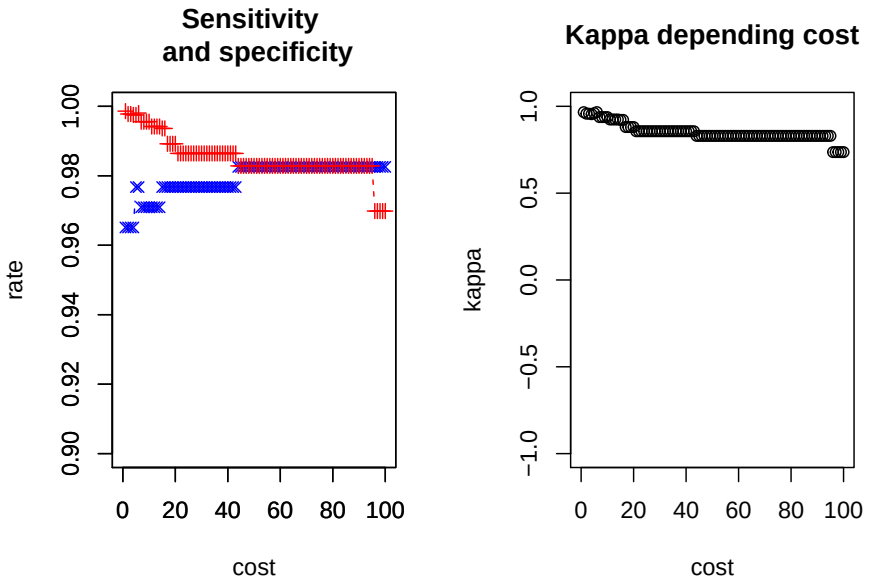
Using a decision tree with a cost matrix give interesting results. A question we ask is : what is the optimal cost to give at FP and FN rate? Intuitively, we want to use the ratio of priors, for example, spotify priors are approximately 0,8 and 0,2 so we use c(1,0) = 1 and c(0,1) = 4, or with recid c(0,1) = 19 (0.95/0.005).

The graphics below shows evolution of sensitivity and specificity in function of the given cost.

The first one confirms that priors can be a good choice to choose the cost for confusion matrix. Globally, CART with cost has improve the predictions in comparison of first models. Not as good as resampling, but it gives substantial improvement. Kappa stills a bit low. A look on the left graph can help users to choose a cost slightly on the right to improove detection power if the cost in specificity is not too expensive for his application of the prediction.



Here, CART gives excellent result, as shows the kappa measure, still high which means results are not due to chance. Beside, TP and TN curves shows that a value between 50 and 90 seems optimal to maintain a high level of detection power and a low level of false alarm and stay balanced between them.

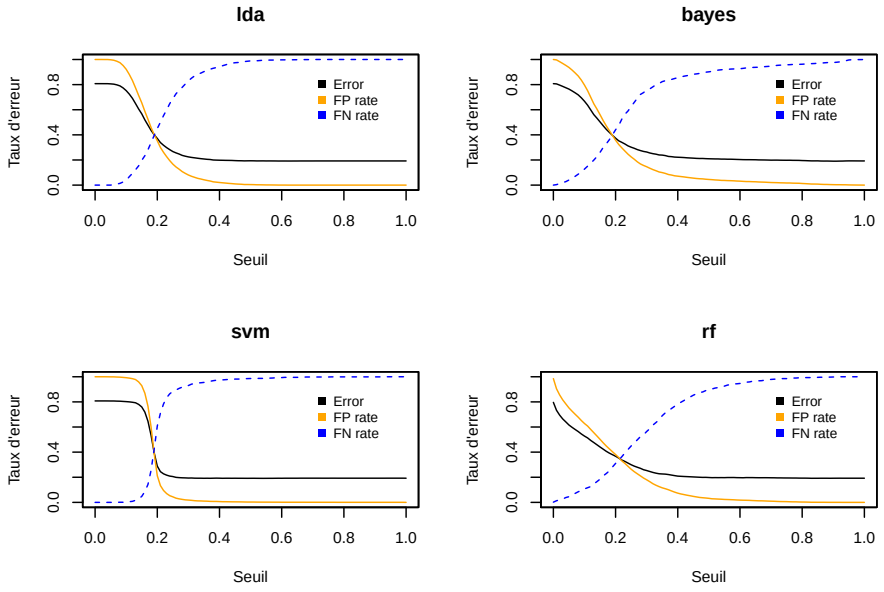


5.5 Post processing Threesholding

In order to choose a good threeshold for the probabilities, we first vizualize the evolution of the three kind of errors. Those graphics will help us to choose a good threshold. We want that global error stays as high as possible. We also want a threeshold where FPrate and FNrate are closed to their intercept. At last we prefer situate on the right of the intercept because we privilegies FPrate to FNrate. the softer the slope is, the easier we can make gains on FP without looing too much in FN and accuracy. Hence, bayes and RF are better here for users who would try different threshold.

Table 5.6: metrics with new threshold

	accuracy	FNrate	TPrate	kappa	PrecisionPPV	Fscore
rf	0.6328511	0.3100917	0.6899083	0.2073542	0.3014028	0.4195258
Bayes	0.6264996	0.440367	0.559633	0.1438748	0.2714731	0.3655978
lda	0.6284404	0.4504587	0.5495413	0.1412709	0.270551	0.3625908
svm	0.7115385	0.6082569	0.3917431	0.1619352	0.305218	0.3431097



Once we have choosen the better value depending our attempts, we can observe results in table 5.6 and reshape with another threshold if necessary.

Once again, we observe a remrquable improvement in comparison of the first try. This threshold can be choosed by someone whants to keep accuracy above 60 percent and keep a balanced between specificity an sensitivity. If the user wants to win some detection power because he has still some margin in terms of specificity and accuracy, he can raise a bit the threshold.

Chapter 6

Summary

- Random forest is globally the best performer. Before the remedies, it is already the one which give on average the best predictions. It reacts very well to pre and post processing manipulations (resampling and thresholding). In several articles, Ensemble classifiers are referred as very efficient in order to counteract imbalanced daataset.(Ribeiro, 2014)
- Results from alternate cutoff are not as good as resampling but they have still relevant results which can be useful for the users who wants shape predictions depending his preference. thresholding can also be combine with resampling. We make the learning with resampled datas, then we shape the results with a slight trehsdolding.
- The cost sensitive learning with CARTS give some significant improvement of the predictions. Even when the first results are already good, this classifier allows to refine through better detection power or false alarm according to he user's choices.
- Resampling works generally very well. However, We notice some significant differences between upsampling and downsizing. Resampling Algorithm based on oversampling seems works better in our works. Among oversampling method, it is hard to say which one is better than another. They give very similar results. It would asked a large study on a large range of data set to really establish a scale of the better "resampling methods" in one or another situation.

6.1 different strategies :

- Data Pre-processing : changes on the data before the learning process takes place
  - advantages:

        * can be applied to any existing tools
        * choosen models are biased to the goals of the users
    - inconevnient:
        * difficult to relate modification of the data with the loss function
- Special-purpose learning method : Modifications of the learning algorithm
    - advantages :
        * users goals are incorporated directly into the models
        * model obtained more comprehensive for the users
    - disadvantages :
        * user is restricted in his algorithmm choices or have to developp new algorithm
        * if the target of the loss function changes, model must be relearned
        * requires deep knowlegde of the learning algorithm implementation
- Prediction Post-processing : transformations applied to the predictions of the learned model
    - advantages :
        * not necessary to know user preference biases at learning time
        * any standard learning tool can be used
    - drawbavks :
        * models do not reflect user preferences
        * models interpretability is meaningless because loss function was not optimized following user preference bias

Bibliography

chawla (2002). *SMOTE: synthetic minority over-sampling technique*. Journal or Artificial Intelligence Research.

Elkan, C. (2002). *The foubdation of cost sensitive learning*.

Japkowicz, N. (2000). *Learning from Imbalanced Data Sets: A Comparison of Various Strategies*. The R journal, DalTech/Dalhousie University, 6050 University.

Kuhn, M. and Jonshon, K. (2013). *Applied Predictive Modeling*. Springer.

Li, C. (2014). *Fisher Linear Discriminant Analysis*. Computer Science.

Ling, , and Li (1998). *Data mining for direct marketing: Problems and solutions*. Proceedings of the Fourth.

Ling, C. X. and Sheng, V. S. (2008). *Cost sensitive learning and the Class Imbalance Problem*. Encyclopedia of Machine Learning, The university of Western Ontario, Canada.

McHugh, M. L. (2015). *Interrater reliability: the kappa statistic*. Biochem Med.

Menardi, N. L. G. and Torelli, N. (2014). *ROSE: A PAckage for Binary Imbalanced*. The R journal.

Ribeiro, B. (2014). *A Survey of Predictive Modelling under Imbalanced Distributions*. Faculdade de Ciencias - Universidade do Porto.

Tabachnick, B. G. and Fidell, L. S. (2007). *Using Multivariate Statistics*. seventh edition.