A Minimal Book Example

Yihui Xie

2021-07-17

Contents

Chapter 1

Introduction

1.1 Abstract

In data-minig, a frequent and major issue for handling with two-class classification is to make reliable predictions with strongly imbalanced distribution of the target variable. Most of the machine learning algorithms assumes by default that all data are balanced. Empirically, a lot of dataset faces this distorsion (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 on 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

On chapter 1, we introduce some tools which allows to measure the effiency of a model.

Chapter 2 briefly presents the differents models we used. We will touch upon the math behind the classifier and the way the algorithm works.

Chapter 3 introduce some remedies to make our classifiers better predictors.

Chapter 4 is an application on three dataset with wich we can evaluate the submitted remedies.

Chapter 5 stands as a conclusion.

Chapter 2

Metrics for classification tasks

Understanding if a model is a good classifier or not require a good comprehension of the predictions he made. We want to know the global effiency but some specific element too. Is he better in predicting one or the other class? What are the strenght and weaknees 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 differents "metrics". The reliability of our tools is higlhy dependant on the structure of our datas. In our case, the imbalanced sample of datas classes has to be take 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 algolritm is globally inneficient. However In the context of imbalanced dataset, 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 personnaly 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 Flase 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 terme “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 litterature 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 combiination of both precision and recall. This metric value is high when both recall (a measure of completness) and precision (a measure of exactness) are high (citation). Hence, this metrics is particuilary 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 (citation).

Kappa Agreement < 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 logistic regression / glmnet

3.3 SVM

3.4 random forest

3.5 Naives bayes

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 repoduce with our level of competence. It appears to us that it is interesting to separate the choosen methods in three levels :

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

4.1 Pre processing resampling

4.2 Learning method tuning

- Metaparameters tuning
- direct sensitive learning

4.3 post processinf threesholding

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.

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 (show plots). Notice that we firs 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 similar results (quote). !! maybe put it in classifiers part !!!!

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 repository, wich contains the .rmd for each dataset. In this repository, 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.

We choose four dataset with different level of imabalanced.

Let's briefly presents those datasets:

- Spotify ...
- Recidivism ...
- Creditcard ...
- Hacked ...

Table of priors ratio between positive and negative class

Table 5.1: Confusion matrix

	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

	accuracy	FNrate	TPrate	kappa	PrecisionPPV	Fscore
rf	0.8020466	0.03211009	0.1055046	0.1032829	0.4389313	0.1701183
Bayes	0.7904023	0.04477938	0.09816514	0.07332293	0.3429487	0.1526391
lda	0.8064573	0.004150284	0.01100917	0.01088917	0.3870968	0.02140946
svm	0.8082216	0.002621232	0.01376147	0.01772557	0.5555556	0.02685765

5.2 First Models

The function models compute our four models. We show the function in order to show the basic parameters. This parameters will be change in a following section. 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.

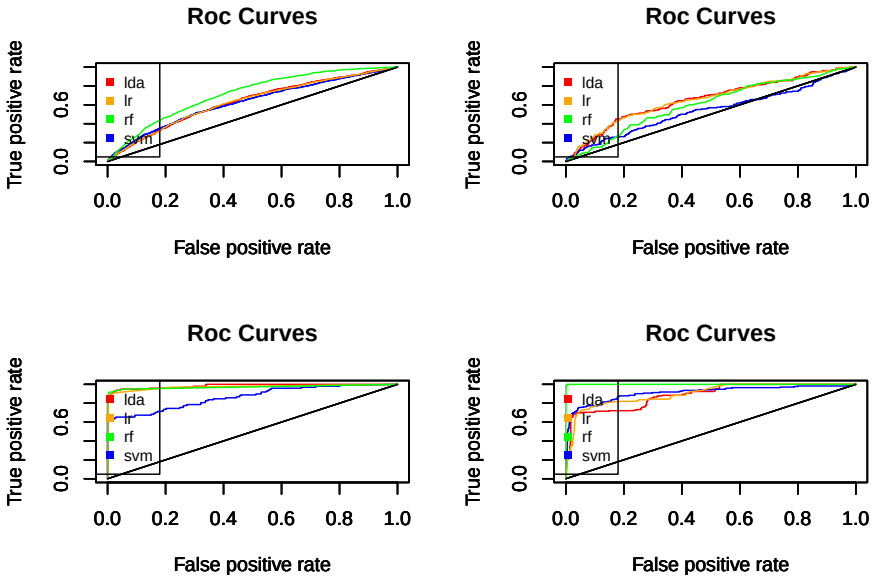
The table … shows the confusion matrix resulting to the four classifiers used on the spotofy dataset. We observe the unabilty to properly predict the unpopular songs. A look on the metrics sharps this observation.

```
KablesPerf(Predict2, datas$test, "popularity")[[2]]
```

First we note that accuracy is very good, wigch confirms accuracy is not a relaible metrics concerning imbalance dataset. a simple view on Detection power(TPr) shows that we don't achieve to predict what songs are very un popular. FN rate is obviously good because of the imblalanced ratio. Here FN won't be a good metrics.

Let see the plot curve for all datasets.

```
par(mfrow = c(2,2))
AllRoc(Predictions, datas$test$popularity)
AllRoc(PredRec1, datRecid$test$is_violent_recid)
AllRoc(PredCredit, creditcard$test$Class)
AllRoc(PredHacked, datHacked$test$MULTIPLE_OFFENSE)
```



!!! discuss the better results for creditcard and hacked !!! - from kaggle (datasets directly from professional use, more relevant, data more reliable) ? preprocessing (pca, onlynumeircal), extreme imbalanced ?

Even if creditcard and hacked seems not to need remedies to counteract imbalanced data, let see the detection power. We can argue that 3/4 of detection power is not enough to reassure users. In a professional use, we can wish at leat 90% of TPr.

Chapter 6

Summary