Imbalanced Binary Classification

Julien Genovese

Machine Learning Together Milan

January 2021

Table of Contents

Introduction to imbalanced binary problems

Definition of imbalanced binary classification problem

- It's a **binary classification problem** where the target output is composed by **two classes where one of them is more frequent than the other one.**

Figure: A classical example of imbalance

- A **very common problem** in real datasets.

Two examples

- **Fraud detection**: in an insurance or financial context we can be interested to distinguish between fraudulent people and onest ones. The first ones are less frequent.
- **Call advertising**: in this case we are interested what kind of people a call center has to contact to have an answer and sell a product. These people are few.

Figure: Two examples of imbalanced classification

Problems with imbalanced datasets

- Most of the algorithms are built under the **assumption of the same number of sample in each class**.

Problems with imbalanced datasets

- Most of the algorithms are built under the **assumption of the same number of sample in each class**.
- Classical metrics to optimize the models will give us apparent good results but not a real good information because they will **focus on the majority class**.

Some solutions

There are different solutions to approach this problem.

- Supervised approach: threshold moving, sampling approach, anomaly detection, ...

Some solutions

There are different solutions to approach this problem.

- Supervised approach: threshold moving, sampling approach, anomaly detection, ...
- Unsupervised approach: isolation forest, ...

Brief mathematical recap on classification

What is classification?

- Classification is a **supervised machine learning problem**.

What is classification?

- Classification is a **supervised machine learning problem**.
- The mathematical model is in the general form of:

$$p = f(X) + \varepsilon$$

where p is the **estimated probability** to belong to a certain class Y, X the **input**, ε an **error term**, f is the **relationship** between the input and the output.

Binary classification

In binary classification:

- We select one of the two classes, that we call the "positive class" and p is the probability to belong to this class

Binary classification

In binary classification:

- We select one of the two classes, that we call the "positive class" and p is the probability to belong to this class
- $1 - p$ is the probability for the other class, the "negative class".

Binary classification

In binary classification:

- We select one of the two classes, that we call the "positive class" and p is the probability to belong to this class
- $1 - p$ is the probability for the other class, the "negative class".
- We classify using the probabilities according to a **threshold**.

Binary classification

In binary classification:

- We select one of the two classes, that we call the "positive class" and p is the probability to belong to this class
- $1 - p$ is the probability for the other class, the "negative class".
- We classify using the probabilities according to a **threshold**.
- **threshold** $= 0.5$ in balanced classification.

Metrics for binary classification

- We want to understand how many events are right predicted, how many mistakes have been done, and which ones.

Metrics for binary classification

- We want to understand how many events are right predicted, how many mistakes have been done, and which ones.
- A classical tool is the **confusion matrix**.

Predicted	Observed	
	Event	Nonevent
Event	TP	FP
Nonevent	FN	TN

Figure: A general confusion matrix

Metrics for binary classification

- We want to understand how many events are right predicted, how many mistakes have been done, and which ones.
- A classical tool is the **confusion matrix**.

Predicted	Observed	
	Event	Nonevent
Event	TP	FP
Nonevent	FN	TN

Figure: A general confusion matrix

- With this matrix we can create different metrics.

Accuracy, Sensitivity, Specificity, Precision

- The **Accuracy rate** is defined as:

$$\frac{TP + TN}{TP + TN + FP + FN}.$$

Accuracy, Sensitivity, Specificity, Precision

- The **Accuracy rate** is defined as:

$$\frac{TP + TN}{TP + TN + FP + FN}.$$

- The **Sensitivity** (or **Recall**) is defined as:

$$\frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}.$$

Accuracy, Sensitivity, Specificity, Precision

- The **Accuracy rate** is defined as:

$$\frac{TP + TN}{TP + TN + FP + FN}.$$

- The **Sensitivity** (or **Recall**) is defined as:

$$\frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}.$$

- The **Specificity** is defined as:

$$\frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FP}}.$$

Accuracy, Sensitivity, Specificity, Precision

- The **Accuracy rate** is defined as:

$$\frac{TP + TN}{TP + TN + FP + FN}.$$

- The **Sensitivity** (or **Recall**) is defined as:

$$\frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}.$$

- The **Specificity** is defined as:

$$\frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FP}}.$$

- The **Precision** is defined as:

$$\frac{\mathrm{TP}}{TP + FP}.$$

The problem we will deal with

Dataframe test

- We will deal with the **TwoClassSim** dataset of the **caret package in R**.
- We have two classes and we have chosen 99.52% of the first class and 0.48% of the second one.

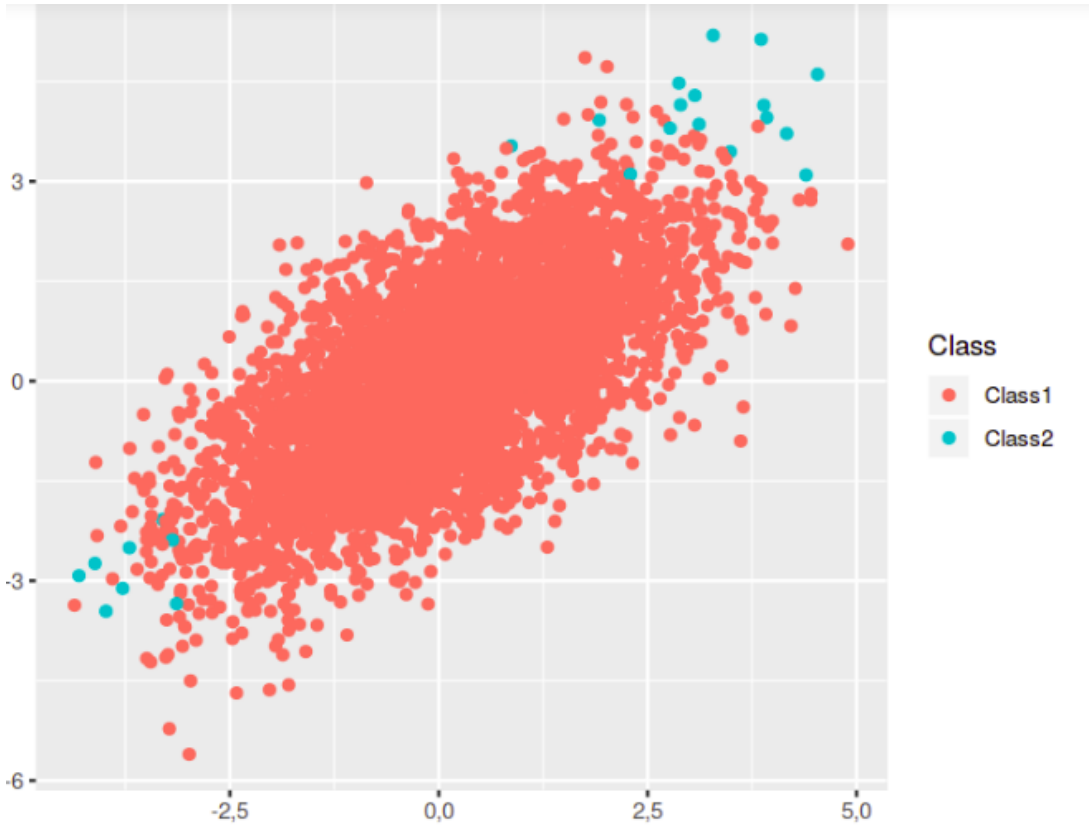


Figure: A projection in 2D of the TwoClassSim dataset

Using classical method for imbalanced problem

- We tried to use a classical approach to this problem optimizing the models w.r.t. the accuracy.

Using classical method for imbalanced problem

- We tried to use a classical approach to this problem optimizing the models w.r.t. the accuracy.
- We choose the less frequent class as positive

Model	Accuracy	Sensitivity	Specificity
Random Forest	0.99	0.26	0.95
Logistic Regression	0.99	0.03	0.99
KNN	0.99	0	1

Using classical method for imbalanced problem

- We tried to use a classical approach to this problem optimizing the models w.r.t. the accuracy.
- We choose the less frequent class as positive

Model	Accuracy	Sensitivity	Specificity
Random Forest	0.99	0.26	0.95
Logistic Regression	0.99	0.03	0.99
KNN	0.99	0	1

- A naive model that predict all the sample as the negative one has an accuracy of 99%.

Using classical method for imbalanced problem

- We tried to use a classical approach to this problem optimizing the models w.r.t. the accuracy.
- We choose the less frequent class as positive

Model	Accuracy	Sensitivity	Specificity
Random Forest	0.99	0.26	0.95
Logistic Regression	0.99	0.03	0.99
KNN	0.99	0	1

- A naive model that predict all the sample as the negative one has an accuracy of 99%.
- It's clear that the models are focusing on the negative class.

Pay attention

- It's up to us to decide if we are more interested in a good positive class classification, in a negative one, or in a trade-off of them.

Pay attention

- It's up to us to decide if we are more interested in a good positive class classification, in a negative one, or in a trade-off of them.
- **The techniques we have to use depends on the problem.**

Supervised method for imbalaced classification

How can we approach the problem?

To deal with the imbalance you can:

- do nothing and cross your fingers.

How can we approach the problem?

To deal with the imbalance you can:

- do nothing and cross your fingers.
- change the classification metric for optimizing.

How can we approach the problem?

To deal with the imbalance you can:

- do nothing and cross your fingers.
- change the classification metric for optimizing.
- change the threshold cutoff to classify.

How can we approach the problem?

To deal with the imbalance you can:

- do nothing and cross your fingers.
- change the classification metric for optimizing.
- change the threshold cutoff to classify.
- use sampling methods.

How can we approach the problem?

To deal with the imbalance you can:

- do nothing and cross your fingers.
- change the classification metric for optimizing.
- change the threshold cutoff to classify.
- use sampling methods.
- weigth the loss for a cost-sentivite training.

Classification metric method

Changing the optimizing metric

- We can chose the best model through the different ones using the **accuracy metric**:

$$\frac{TP + TN}{TP + TN + FP + FN}.$$

Changing the optimizing metric

- We can chose the best model through the different ones using the **accuracy metric**:

$$\frac{TP + TN}{TP + TN + FP + FN}.$$

- Problem: this metric is **too much oriented to the most frequent class** due to the difference of order of magnitude.

Changing the optimizing metric

- We can chose the best model through the different ones using the **accuracy metric**:
$$\frac{TP + TN}{TP + TN + FP + FN}.$$
- Problem: this metric is **too much oriented to the most frequent class** due to the difference of order of magnitude.
- Solution: we can optimize w.r.t. sensitivity, precision, AUC ROC curve, AUC precision-recall curve.

ROC curve

- We have $1 -$ Specificity on the $x-$ axis and Sensitivity on the y-axis.

$$1 - \text{Specificity} = 1 - \frac{\text{TN}}{\text{TN+FP}} = \frac{\text{FP}}{\text{TN+FP}}.$$

ROC curve

- We have $1 - \text{Specificity}$ on the $x-$ axis and Sensitivity on the y-axis.

$$1 - \text{Specificity} = 1 - \frac{\text{TN}}{\text{TN+FP}} = \frac{\text{FP}}{\text{TN+FP}}.$$

- The ROC curve is created changing the threshold for the model and predicting the associated label. After that we measure for each threshold/prediction step the Sensitivity and $1 - \text{Specificity}$.

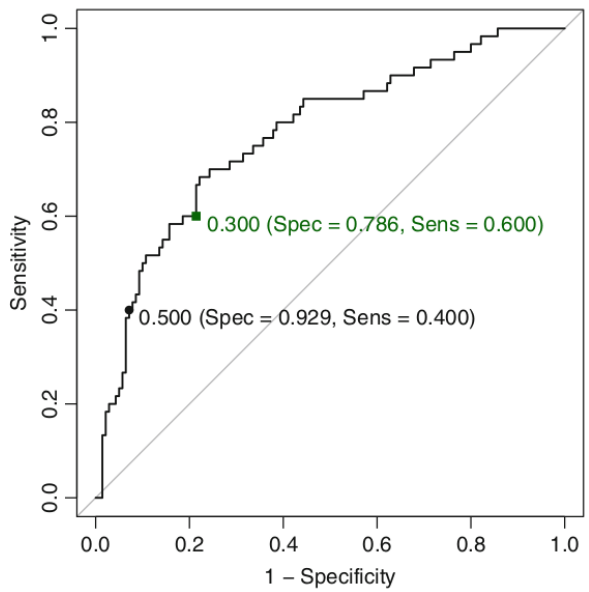


Figure: ROC curve of a classifier against random guess

Some observations on the ROC curve

- The **bisector** is the ROC curve for the **random guessing**.

Some observations on the ROC curve

- The **bisector** is the ROC curve for the **random guessing**.
- A **perfect classifier** has a ROC curve of the type **y=1**.

Some observations on the ROC curve

- The **bisector** is the ROC curve for the **random guessing**.
- A **perfect classifier** has a ROC curve of the type **y=1**.
- We can use the **AUC of the ROC curve to select a model**.
  $\Rightarrow$ threshold-insensitive.
  $\Rightarrow$ insensitive to disparity in the class proportions (it's a function of sensitivity and specificity).

Problems with AUC ROC curve

Problems:

- different **curves could cross**.

Problems with AUC ROC curve

Problems:

- different **curves could cross**.
- the **threshold** can be an **important discriminating factor**.

Problems with AUC ROC curve

Problems:

- different **curves could cross**.
- the **threshold** can be an **important discriminating factor**.
- with very **imbalanced dataset is not reliable**.

Example of ROC problem

Let's see an example of the previous problem.

Predicted — Reference	Negative	Positive
Negative	9.4e+04	10
Positive	4.4e+03	1.6e+02

In this case we have:

- Sensitivity: 0.94
- Specificity: 0.95
- Precision: 0.035

The ROC Problem

- This is related to the fact that:

$$FPR = 1 - \text{Specificity} = \frac{FP}{TN + FP}$$

  and FP can be very different in order of magnitude with respect to TN.
  $\Rightarrow$ we can have a FP number high w.r.t. TP but low w.r.t. TN.
  $\Rightarrow$ the FPR becames always near to zero
  $\Rightarrow$ the ROC curve seems similar to that one of the perfect classifier
- we can have a very good recall and sensitivity and low precision.

Precision-Recall Curve

- According to the precision-recall trade-off we can also construct the precision-recall curve.

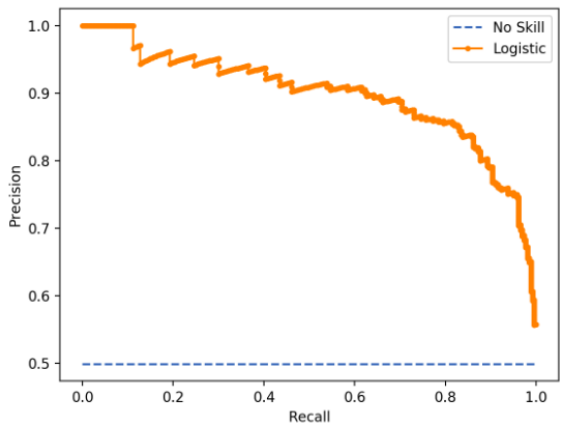


Figure: Precision-Recall curve

Precision-Recall Curve

- According to the precision-recall trade-off we can also construct the precision-recall curve.

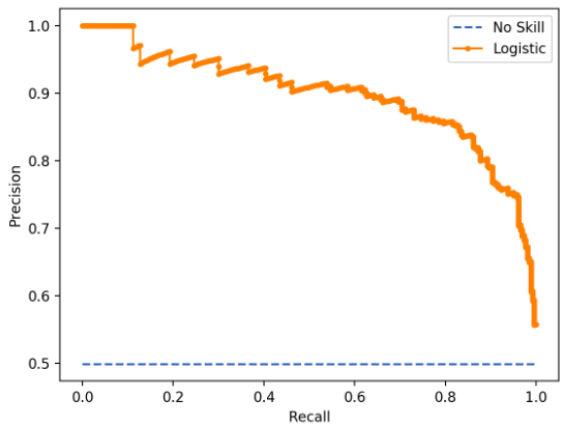


Figure: Precision-Recall curve

- In this case too we can **select the best model** according to the **AUC of the precision-recall curve**.

Precision-Recall Curve

- According to the precision-recall trade-off we can also construct the precision-recall curve.

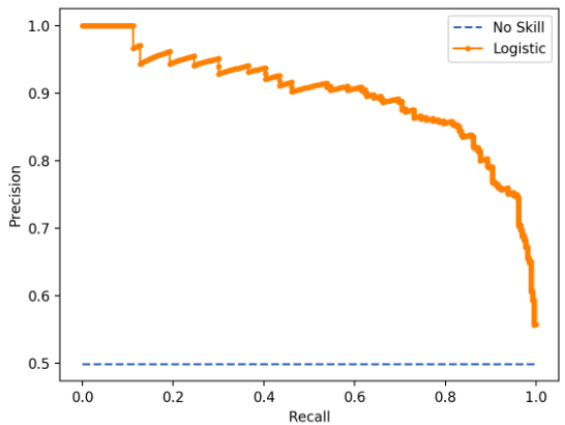


Figure: Precision-Recall curve

- In this case too we can **select the best model** according to the **AUC of the precision-recall curve**.
- In a **very imbalanced dataset** (e.g 1:100) it's **better to use the precision-recall curve** because the ROC is optimistic.

Numerical example

Let's try to optimize a random forest w.r.t. the ROC:

mtry	ROC	Sens	Spec	Accuracy
2	0.95	0	1	0.995
7	0.949	0	1	0.995
13	0.97	0.13	0.999	0.995
19	0.95	0.26	0.999	0.996
25	0.944	0.23	0.999	0.995
31	0.946	0.33	0.999	0.996
37	0.947	0.35	0.999	0.995
43	0.948	0.33	0.999	0.996
49	0.925	0.38	0.998	0.996
55	0.942	0.38	0.998	0.996

Threshold selection method

Threshold selection

- Having the probabilities we can **change the threshold to predict the associated labels**.

Threshold selection

- Having the probabilities we can **change the threshold to predict the associated labels**.
- Changing the threshold **we change sensitivity, specificitity, precision** and so on.

Threshold selection

- Having the probabilities we can **change the threshold to predict the associated labels**.
- Changing the threshold **we change sensitivity, specificitity, precision** and so on.
- We can use both the **ROC curve and the precision recall**.

Select the threshold using a ROC curve

- Searching for the point nearest to the point with 100% Sensitivity and 100% Specificity (the top left corner) and chose the associated threshold.

Select the threshold using a ROC curve

- Searching for the point nearest to the point with 100% Sensitivity and 100% Specificity (the top left corner) and chose the associated threshold.
- Searching the threshold that maximizes the **Youden's J index**:

$$\mathbf{J}(thr) = Sensitivity(thr) + Specificity(thr) - 1$$

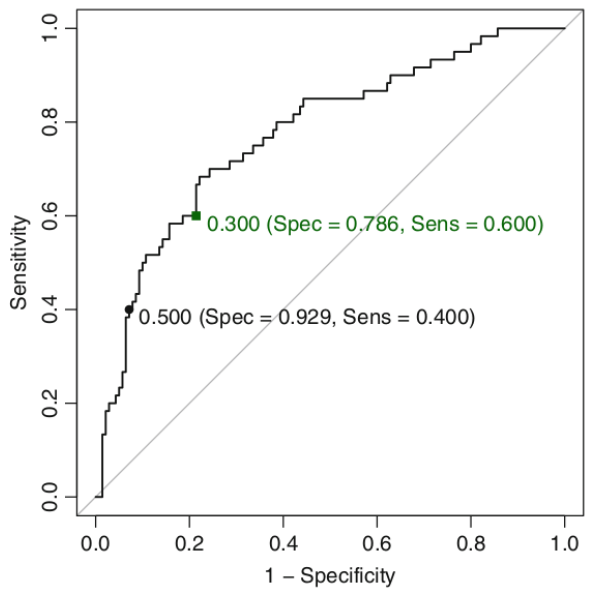
Select the threshold using a ROC curve

- Searching for the point nearest to the point with 100% Sensitivity and 100% Specificity (the top left corner) and chose the associated threshold.
- Searching the threshold that maximizes the **Youden's J index**:

$$\mathbf{J}(thr) = Sensitivity(thr) + Specificity(thr) - 1$$

- Searching the threshold that maximizes the **G-mean**:

$$\mathbf{G} = \sqrt{Sensitivity(thr) \cdot Specitificity(thr)}$$



Numerical example

We use a r.f. with $mtry = 13$ changing the threshold:

Method	Threshold	Specificity	Sensitivity
Default	0.5	0.999	0.133
Clos.topleft	0.047	0.98	0.96
Youden	0.047	0.98	0.96

Select the threshold using a precision-recall curve

- Searching for the point nearest to the point with 100% Precision and 100% Recall (the top right corner) and chose the associated threshold.

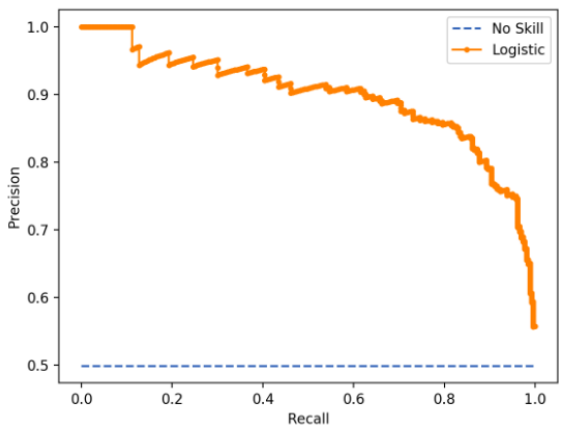


Figure: Precision-Recall curve

Select the threshold using a precision-recall curve

- Searching for the point nearest to the point with 100% Precision and 100% Recall (the top right corner) and chose the associated threshold.
- Searching the threshold that maximizes the **F1-score**:

$$\mathbf{F}_1(thr) = \frac{2}{\frac{1}{\text{precision}(thr)} + \frac{1}{\text{recall}(thr)}}$$

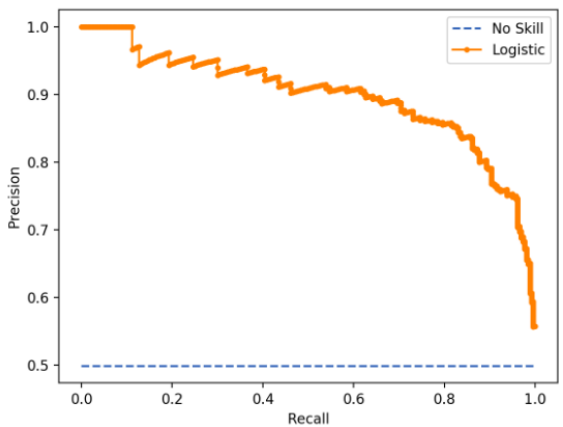


Figure: Precision-Recall curve

How to select the threshold

The threshold can be seen as a model parameter.
To select it we have to use:

- A **training set** for learning the different model and select the best one (using a threshold invariant metric)

How to select the threshold

The threshold can be seen as a model parameter.
To select it we have to use:

- A **training set** for learning the different model and select the best one (using a threshold invariant metric)
- An **evaluation set** to select the threshold.

How to select the threshold

The threshold can be seen as a model parameter.
To select it we have to use:

- A **training set** for learning the different model and select the best one (using a threshold invariant metric)
- An **evaluation set** to select the threshold.
- A **test set** to have un unbiased estimation of the error.

Pay attention

- The core model doesn't change changing the threshold!

Pay attention

- The core model doesn't change changing the threshold!
- We only change the trade-off between particular type of errors.

Pay attention

- The core model doesn't change changing the threshold!
- We only change the trade-off between particular type of errors.
- In a confusion matrix we are moving up and down rows of the matrix and not moving from the off-diagonal to the diagonal.

Pay attention

- The core model doesn't change changing the threshold!
- We only change the trade-off between particular type of errors.
- In a confusion matrix we are moving up and down rows of the matrix and not moving from the off-diagonal to the diagonal.
- We are not inducing further separation between the classes.

Sampling methods

Different techniques to solve class imbalance:

- **Priori** technique: balance sampling more the less frequent class

Different techniques to solve class imbalance:

- **Priori** technique: balance sampling more the less frequent class
- **Post** technique:

Different techniques to solve class imbalance:

- **Priori** technique: balance sampling more the less frequent class
- **Post** technique:
  - **Up-sampling**

Different techniques to solve class imbalance:

- **Priori** technique: balance sampling more the less frequent class
- **Post** technique:
  - **Up-sampling**
  - **Down-sampling**

Different techniques to solve class imbalance:

- **Priori** technique: balance sampling more the less frequent class
- **Post** technique:
  - **Up-sampling**
  - **Down-sampling**
  - **SMOTE**

Priori sampling

- We try to balance sampling the less frequent class at priori.

Priori sampling

- We try to balance sampling the less frequent class at priori.
- The training set $\Rightarrow$ can be balanced.

Priori sampling

- We try to balance sampling the less frequent class at priori.
- The training set $\Rightarrow$ can be balanced.
- The test set $\Rightarrow$ must reflect the imbalance of the real data.

Priori sampling

- We try to balance sampling the less frequent class at priori.
- The training set $\Rightarrow$ can be balanced.
- The test set $\Rightarrow$ must reflect the imbalance of the real data.
- Sometimes is not possible to obtain more data.

Post hoc sampling: Up-sampling

- We **duplicate some of the data in the minority class.**

Post hoc sampling: Up-sampling

- We **duplicate some of the data in the minority class**.
- We use a sampling with replacement until we don't reach the same number of sample in each class.

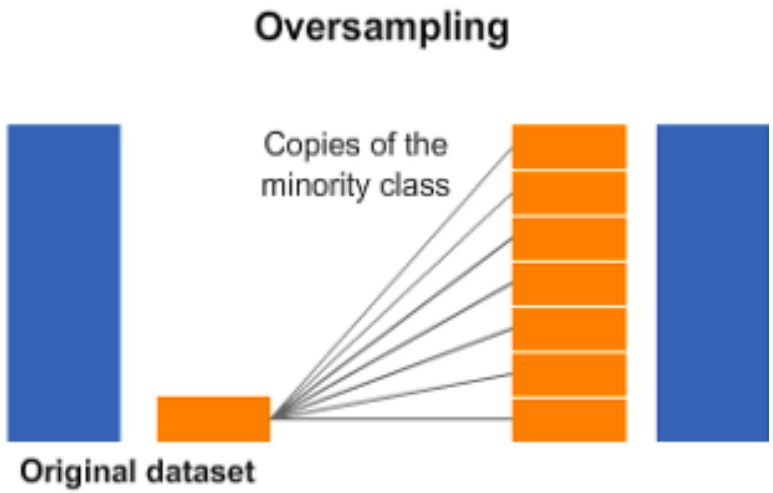


Figure: Upsampling Proceduce

Post hoc sampling: Up-sampling

- We **duplicate some of the data in the minority class**.
- We use a sampling with replacement until we don't reach the same number of sample in each class.

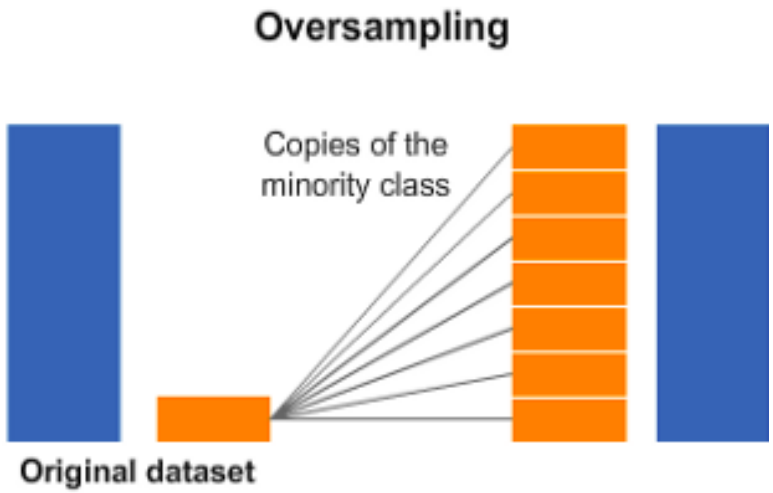


Figure: Upsampling Proceduce

- We don't lose informations but we decrease the variability of our data.

Post hoc sampling: Down-sampling

- We **remove some of the data in the majority class.**

Post hoc sampling: Down-sampling

- We **remove some of the data in the majority class**.
- We remove at random until we don't reach the same number of sample in each class.

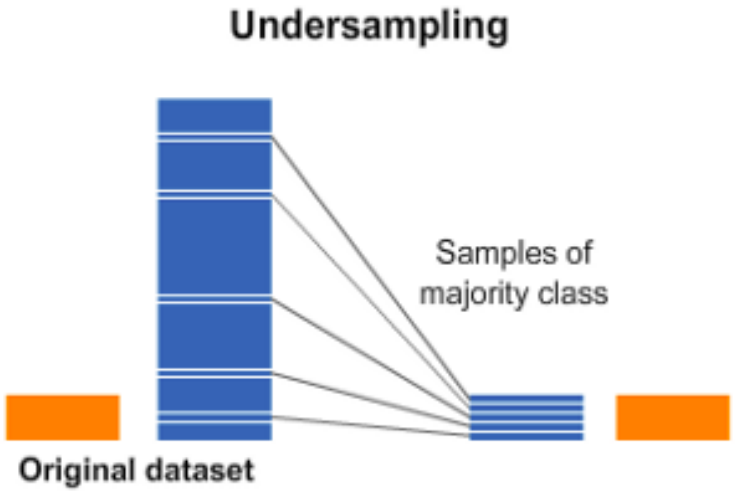


Figure: Downsampling Proceduce

Post hoc sampling: Down-sampling

- We **remove some of the data in the majority class**.
- We remove at random until we don't reach the same number of sample in each class.

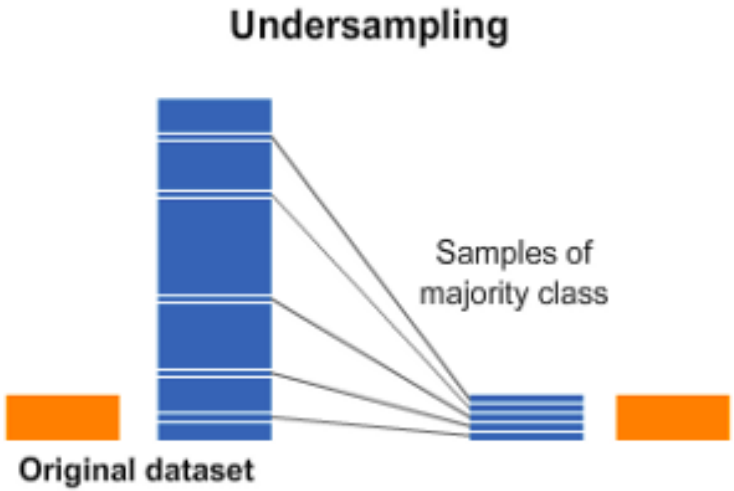


Figure: Downsampling Proceduce

- We lose informations but we don't have big effect on the variability of our data.

Post hoc sampling: SMOTE

- Use both a down sampling and new up sampling.

Post hoc sampling: SMOTE

- Use both a down sampling and new up sampling.
- The down-sampling is the previous one.

Post hoc sampling: SMOTE

- Use both a down sampling and new up sampling.
- The down-sampling is the previous one.
- The up one is a Syntetic generation data using a KNN approach.

Post hoc sampling: SMOTE

- Use both a down sampling and new up sampling.
- The down-sampling is the previous one.
- The up one is a Syntetic generation data using a KNN approach.
- The generated sample is a random combination of randomly selected data points.

SMOTE algorithm: downsampling phase

- Randomly remove sample from majority class population until minority class becomes some specified percentage of minority class.

SMOTE algorithm: downsampling phase

- Randomly remove sample from majority class population until minority class becomes some specified percentage of minority class.
- e.g.:
  - majority class: 200 sample
  - minority class: 50 sample
  - Percentage we down: 200%.

  $\Rightarrow$ majority class: 25 sample.

SMOTE algorithm: downsampling phase

- Randomly remove sample from majority class population until minority class becomes some specified percentage of minority class.
- e.g.:
  - majority class: 200 sample
  - minority class: 50 sample
  - Percentage we down: 200%.

  $\Rightarrow$ majority class: 25 sample.
- Percentage of downsample of majority class $\Rightarrow$ Parameter to tune.

SMOTE algorithm: syntetic generation

- Select a parameter k to define the **number of neighbors** to take.

SMOTE algorithm: syntetic generation

- Select a parameter k to define the **number of neighbors** to take.
- Select a **percentage of the minority class** to generate.

SMOTE algorithm: syntetic generation

- Select a parameter k to define the **number of neighbors** to take.
- Select a **percentage of the minority class** to generate.
- **For each point of the minority class** take the k neighbors and according to the percentage to generate **take random neighbors from the k ones**

SMOTE algorithm: syntetic generation

- Select a parameter k to define the **number of neighbors** to take.
- Select a **percentage of the minority class** to generate.
- **For each point of the minority class** take the k neighbors and according to the percentage to generate **take random neighbors from the k ones**
- For each point take a random **convex combination** between the point and each neighbors.

SMOTE algorithm: syntetic generation

- Select a parameter k to define the **number of neighbors** to take.
- Select a **percentage of the minority class** to generate.
- **For each point of the minority class** take the k neighbors and according to the percentage to generate **take random neighbors from the k ones**
- For each point take a random **convex combination** between the point and each neighbors.
- The point obtained with the combination is the **new sintetic generated one**.

When do we have to sample?

Different moments when we can use sampling methods:

- Before the cross validation
  $\Rightarrow$ the dev error is biased.
  $\Rightarrow$ less computational cost.

When do we have to sample?

Different moments when we can use sampling methods:

- Before the cross validation
  $\Rightarrow$ the dev error is biased.
  $\Rightarrow$ less computational cost.
- During the cross validation
  $\Rightarrow$ the dev error is not biased.
  $\Rightarrow$ more computational cost.

Numerical example

We tried to compare the three sampling methods with a logistic regression:

Method	ROC	Sensitivity	Specificity	Accuracy
No sampling	0,759	0,05	0,998	0,99
Up	0,62	0,65	0,56	0,56
Down	0,62	0,65	0,56	0,56
SMOTE	0,63	0,53	0,757	0,757

Pay attention

- We have to apply the **post sampling only on the training set** to better learn.

Pay attention

- We have to apply the **post sampling only on the training set** to better learn.
- The **test set** must follow the **original distribution**.

Pay attention

- We have to apply the **post sampling only on the training set** to better learn.
- The **test set** must follow the **original distribution**.
- Using a sampling method on the test set will result in a **biased error**.

Cost-sensitive training

Introducing a new loss

- We can optimize a modified loss function instead of the classical one.

Introducing a new loss

- We can optimize a modified loss function instead of the classical one.
- The new loss weights the two classes differently.

Introducing a new loss

- We can optimize a modified loss function instead of the classical one.
- The new loss weights the two classes differently.
- This affect the parameters and can have significant improvements.

Some examples

- Loss for logistic regression:

$$L = \sum_{i=1}^{n} w_i \cdot y_i \cdot \log \hat{y}_i$$

Some examples

- Loss for logistic regression:

$$L = \sum_{i=1}^{n} w_i \cdot y_i \cdot \log \hat{y}_i$$

- Gini index for a tree:

$$Gini = C(1|2)p_1(1-p_1) + C(2|1)p_2(1-p_2)$$

where $C(1|2)$ and $C(2|1)$ are the weights of missclassification.

Some examples

- Loss for logistic regression:

$$L = \sum_{i=1}^{n} w_i \cdot y_i \cdot \log \hat{y}_i$$

- Gini index for a tree:

$$Gini = C(1|2)p_1(1-p_1) + C(2|1)p_2(1-p_2)$$

  where $C(1|2)$ and $C(2|1)$ are the weights of missclassification.
- We cannot have probabilities. They makes no sense. We cannot optimize w.r.t. ROC

A numerical example

We use a weighted loss for our example:

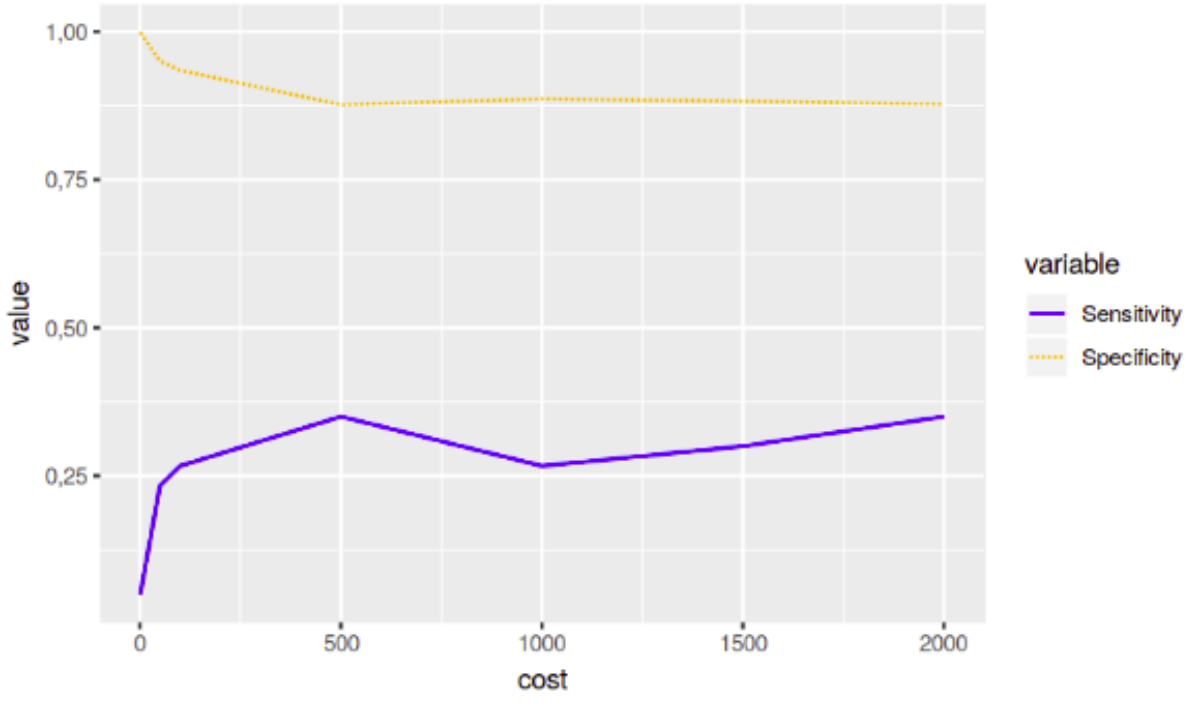


Figure: Sensitivity and Specificity against the weight

The Bibliography

[1] Max Kuhn, Kjell Johnson: *Applied Predictive Modeling*. Springer, 2016.

[2] Tom Fawcett: *Learning from Imbalanced dataset*. Link to the article.

[3] Wicked Good Data - r: *Handling class imbalance with R and Caret - An introduction*. Link to the article

[4] Jason Brownlee: *A Gentle Introduction to Imbalanced Classification*. Link to the article.

[5] LCT14558: *Imbalanced data and why you should NOT use ROC curve* Link to the notebook.

[6] Jason Brownlee: *A Gentle Introduction to Threshold-Moving for Imbalanced Classification*. Link to the article.

[7] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, W. Philip Kegelmeyer: *SMOTE: Synthetic Minority Over-sampling Technique*. Journal of Artificial Intelligence Research 16 (2002) 321–357

[8] Jason Brownlee: *SMOTE for Imbalanced Classification with Python* Link to the article.

Thank you for your attention!!