UNIVERSITY OF BARCELONA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

Course project: A Reappraisal of Bayesian Classification with Unbalanced Data

Student name: *Adrián Fernández Cid, Aitor Lucas Castellano, Marcos Moreno Blanco & Noel Rabella Gras*

Course: *Bayesian Statistics and Probabilistic Programming*
Due date: *June 28th, 2021*

Abstract

In this work we revisit a known binary classification problem with unbalanced data, the Kaggle challenge *Any boson at home?*[1], to expand on Bayesian solutions in such scenarios as a continuation to the analysis performed in the second assignment of the course. Specifically, we apply two more sophisticated undersampling strategies to try and draw the most informative samples for inference, that we call Minimal Targetted Removal (MTR) and Recursive Targetted Removal (RTR); and implement a spike-and-slab regression. Results show a moderately positive effect of MTR on recall, while RTR appears to yield overfitting and the spike-and-slab prior bears no significant effect. Furthermore, the absolute test AUCs obtained across all cases studied are below the top results of the challenge, which we attribute to our logistic regression not being able to capture the non-linearity of the decision boundary.

1. Introduction

Upon finishing the second assignment, we were left eager to try new ideas we did not have the time to implement, namely a more sophisticated undersampling strategy and models other than the logistic regression. This, added to the facts that we had been pleased to work on a typical machine learning problem such as the one proposed for that assignment and that the topic of imbalance seemed relevant to us (as we have often encountered the issue before), led us to choose a similar problem for our project. This has allowed us to deepen our understanding of a typical data science problem (rather than acquiring only a superficial grasp of a completely novel one), pushing us to exercise what we have learnt in the course beyond the sheer contents thereof (as the undersampling techniques we use and the spike-and-slab model have not been directly addressed in class).

Furthermore, we have been fortunate to know of a candidate problem beforehand: the Kaggle challenge *Any boson at home?*[2], which consists on identifying a particular decay process in a dataset containing particle collisions produced at the LHCb experiment at CERN. Not only is it appropriate because it is an unbalanced binary classification

[1]See https://www.kaggle.com/c/cernsignal/overview.
[2]See previous footnote.

problem (as the one in the second assignment), but also because we have worked on it in the Machine Learning course, which will allow us to benchmark the Bayesian models used (as in the Machine Learning course we limited our treatment to frequentist models) and to concentrate on the Bayesian analysis rather than on the specific content of the problem. In this regard, we focus on gauging the benefit of the undersampling techniques used and the spike-and-slab model,, instead of on obtaining the best performance by any means.

The rest of the report is structured as follows. Below we introduce the problem at hand. Subsequently, we present the strategy we followed to solve it. Then, the next two sections summarise the basis for the new undersampling techniques used and the spike-and-slab model, respetively. We present and discuss our results in the following section, and conclude with some final remarks.

For the sake of concision, we have limited this report to what we considered most relevant, *i.e.* novel methods and information with respect to the previous assignment. We refer to the notebooks we provide for details on the logistic regression, the horseshoe prior and other previously established techniques.

2. The problem

We have a tabular dataset with information on particle collisions recorded at the LHCb experiment at CERN[3] (Switzerland), and the objective is to identify the presence of a specific process: *the decomposition of B_0 into K^0*. We therefore have a binary classification problem, in which the focus is placed on the pòsitive signal. This last point suggests we keep an eye on recall (also known as sensitivity) to maximise true postives.

Since the meaning of the predictors is not obvious, we include their description here:

- **B_FDCHI2_OWNPV**. Flight distance of the B meson with respect to the primary vertex (point of proton-proton collision).
- **B_IPCHI2_OWNPV**. Impact parameter. Shortest distance between the B meson trajectory and the primary vertex.
- **B_PT**. B meson transverse momentum
- **PIMINUS** & **KPLUS PSEUDORAPIDITY**. Angle of the piminus and Kplus particles with respect to the beam axis.
- **KPLUS** & **PIMINUS P**. Kplus and piminus momentum.
- **KST_892_0_COSTHETAH**. K* helicity angle. Angle formed bewteen one of the particle products of K* (piminus or Kplus) and the B meson in the reference frame of the K* (rest frame)
- **GAMMA_PT**: photon transverse momentum.

[3]The LHCb is a particle physics experiment whose main purpose is to study the phenomenon of CP violation produced in the interactions of particles known as b-hadrons (hence the name, "Large Hadron Collider beauty"), aiming to explain the imbalance between matter and antimatter observed in the universe. See https://en.wikipedia.org/wiki/LHCb_experiment for more details.

- **KST_892_0_IP_OWNPV**. Impact parameter of K*, which corresponds to the shortest distance between the K* trajectory and the primary vertex.
- **B_OWNPV_CHI2**. Chi2 test of the primary vertex distribution.
- **KPLUS** & **PIMINUS IP_OWNPV**. Shortest distance between the Kplus or the piminus trajectory and the primary vertex.
- **B_DIRA_OWNPV**. Corresponds to the cosine of the angle formed by the momentum of the B meson and the vector formed between the initial (primary vertex) and final position (decay vertex) of the B meson.

3. Strategy

The procedure we followed consisted of various steps. First, we defined our baseline as a logistic regression with random undersampling and Gaussian (that we also call "vanilla") priors for the logistic parameters β_i (we also tried horseshoe priors with no improvement in performance, so we decided to drop them for the rest of the analysis): this is done in `CERN-RandomSamples-Logistic-JAGS.ipynb`.

Then, we added to the baseline a first, timid (and simple) approach to targetted undersampling: what we call *Minimal Targetted Removal* (MTR, in `CERN-MinTargSamples-Logistic-JAGS.ipynb`). Such an approach consists of a one-step targetted removal (either with One-Sided Selection or the Neighbourhood Cleaning Rule; see following section) followed by a random trimming of the resulting subsamples to obtain an affordable size for JAGS. Subsequently, we implemented a *Recursive Targetted Removal* (RTR, in `CERN-OssSamples-Logistic-JAGS.ipynb` and `CERN-NcrSamples-Logistic-JAGS.ipynb`), which attempts to maximise the number of samples judiciously (*i.e.* with OSS or NCR, instead of randomly) selected/discarded by alternating between targetted and random removal: this method is also explained in the below section. Finally, once we had tested both the MTR and RTR approaches, we implemented the spike-and-slab model (see section 5) in `CERN-RandomSamples-Logistic-SpikeSlab.ipynb`: since neither of the two targetted removal approaches we adopted seemed to contribute significantly to improving performance, we used only random undersampling in this last case.

On a more general note, since we are forced to strongly undersample the data (down to approximately 2000 samples), we take three different subsamples in each case (each with a different, given random seed) to check for consistency in the results. Such is the case with all implementations except for that of the spike-and-slab prior, which is sufficiently fast to allow a greater train size and thereby not require (at least as much as the rest) taking several independent subsamples.

Moreover, while we keep track of the typical statistical quantities for evaluating performance (deviance, AIC and BIC) we introduce and focus on the AUC and recall measures. The reasons for this are: i) that AUC and recall are more associated to machine learning classification problems, such as the one considered here; ii) that both measures are bounded (they each lie between 0 and 1) and so provide an absolute notion of performance, contrary to deviance, AIC and BIC which become arbitrarily large as the size of the dataset increases; and iii) that recall (the number of positives predicted as such) is particularly suited for problems where interest lies on the positive signal, as is our case[4].

Finally, as an exercise and for completion, we have included the Stan implementation of the JAGS cases with random and recursive OSS undersampling (in `CERN-RandomSamples-Logistic-Stan.ipynb` and `CERN-OssSamples-Logistic-Stan.ipynb`): the rest of the Stan versions can be obtained from those and the JAGS counterparts without difficulty.

4. On undersampling techniques

We owe much of our exploration of undersampling options to a blog post by Jason Brownlee[5]; we summarise our findings here.

As we have mentioned, random undersampling may discard relevant (informative) samples and keep redundant, interior (far from the boundary) or ambiguous (noisy) ones. To address this issue, two different and complementary paths open quite naturally: one is to develop methods to identify *what to keep* from the majority class (informative, non-redundant samples), while the second focuses on finding *what to remove* (noisy data).

In the first category we have the Near-Miss (NM) and Condensed-Nearest-Neighbours (CNN) algorithms. The NM rule [Zhang and Mani(2003)] comprises 3 different strategies, all involving the distance of majority class examples to minority class ones:

- NM-1. Keep only majority samples with minimum average (usually Euclidean) distance to the three closest minority points. On the toy dataset used by Brownlee, this method keeps samples in overlapping regions, in a roughly distributed way.
- NM-2. Keep only majority samples with minimum average distance to the three furthest minority points. This method appears to retain points in a cluster in the overlapping region, which is an undesirable property because it only preserves information on a localised part of the boundary.
- NM-3. Keep only majority samples with minimum average distance to all the minority points. Contrary to the previous one, this variant preserves samples all over the overlapping region (even more distributed than with NM-1), which means it better captures the whole boundary.

Usually, one applies either of the three strategies until the resulting subset is deemed sufficiently balanced.

The CNN method [Hart(1968)] is based on the nearest-neighbours (KNN) logic. One starts with a set of samples ("store"), which for unbalanced problems usually consists

[4]Recall is even more relevant when the signal of interest is under-represented in the data, since *e.g.* a classifier predicting only the majority class will yield an 80% accuracy on a dataset where the positive signal is present only in 20% of the samples. This, however, will not be particularly relevant in our case, since we balance the data prior to modelling.

[5]See https://machinelearningmastery.com/undersampling-algorithms-for-imbalanced-classification/.

of the whole minority class and possibly a seed of one or a few majority samples, and progressively adds new samples only if they are misclassified by the KNN evaluation of the current store. One immediate consequence of the algorithm is that majority samples around minority ones tend to be retained. Additionally, this method does not generally provide a perfectly balanced subset, but such a feature is positive to the extent that excess majority points are relevant (as, in general, one can have a problem with more informative samples of one class than of another, and the optimal result is to be expected from a maximal number of informative points). There is, however, variability in the seed of initial majority samples to consider (both its size and the actual points chosen) and in the order of evaluation of the rest: one point may be misclassified if evaluated at the beginning, while it could have been correctly classified if evaluated later, with a larger sample store. This means that CNN is prone to including early-considered redundant samples, which adds to the fact that the majority samples surrounded by minority ones it naturally keeps might constitute just noise, instead of boundary-relevant points (although this is a more general problem and it also affects the NM method). Furthermore, the method is rather slow, which favours small datasets and low k values.

We have mentioned above that close cross-class samples may indicate the true boundary, but may also constitute noise adding spurious ambiguity to it. This is a difficult problem, as there is no immediate way to determine the relevance of such samples without already knowing the boundary. NM and CNN risk adding too many of these and in return are more certain to keep boundary-relevant points; the Tomek Links (TM) and Edited Nearest Neighbour (ENN) methods take the opposite approach, *i.e.* removing potentially noisy samples. These are thus strategies to decide what samples to discard.

A Tomek link [Tomek(1976a)] is formed by two different-class samples that are each other's nearest neighbour. Identifying such links and removing the majority-class members therefore reduces noise and simplifies the boundary. While removing noise can be expected to improve generalisation performance, simplifying the boundary by removing potentially informative points might be ill-advised. Nevertheless, such an approximation may prove useful with linear classifiers (such as the logistic regression used in this project).

Another method used to "clean" the data is Edited Nearest Neighbours [Wilson(1972)]. In a way, this is the opposite of the CNN method: instead of keeping misclassified samples, it discards them. In particular, when working with unbalanced data, it evaluates the whole dataset with a 3NN model and removes all misclassified majority samples. As pointed out by Tomek [Tomek(1976b)], this approach can be extended by applying the method recursively until having a perfectly classifiable subset, and/or considering several different k (instead of just 3).

Both the TL and ENN algorithms aim to reduce noise, not to solve imbalance: they will generally remove only a few samples. Their implementation for unbalanced problems is thus most profitable when combined with that of NM or CNN (that discard redundant samples, leaving a more balanced dataset). Two such combinations are One-Sided Selection (OSS) and the Neighbourhood Cleaning Rule (NCR). OSS [Kubat and Matwin(1997)] combines the CNN and TL methods, while NCR [Laurikkala(2001)] complements ENN by also removing all majority samples in the 3-neighbourhood of any misclassified minority sample.

Fortunately, the R package `unbalanced` [A. Dal Pozzolo(2015)] provides all the above methods except for NM. The best option for us is likely a combination of denoising (to simplify the boundary for our linear classifier) and redundancy removal (to balance the data as much as possible, since we are most interested in the positive signal), so both OSS and NCR look like worthy candidates.

Furthermore, we recall that our particular problem is not just imbalance, but also an upper bound on the size of the dataset used (due to the lengthy evaluation of MCMC methods). This means we need to remove a minimum number of points (we have a dataset of around 210 000 samples, and 2000 samples take us approximately 40 min to process with JAGS, which we deem a reasonable maximum execution time), so we have three options:

1. Apply either of the chosen strategies recursively until we reach an acceptable sample size (possibly with some constraint to assure final balance).
2. Apply targetted undersampling once, then randomly trim the result to fit our computational needs. We call this Minimal Targetted Removal (MTR).
3. Try to tweak the parameters of the pre-defined functions to make them reduce the dataset to a workable version without the need for recurrence nor random (*i.e.* blind) undersampling.

Given that the undersampling methods of the `unbalanced` package remove progressively fewer samples when approaching balance (this seems a feature of the algorithms themselves, rather than of their implementation in the package), the first option is barred. Neither have we been able to tinker much with the parameters of the functions, as they leave little to no freedom regarding the inner structure of the algorithm. Therefore, looks like we finally have to accept some random undersampling: as we have announced in section 3, we one of the undersampling implementations we did is the MTR.

However, since we would like to maximise the number of samples selected/discarded by the more sophisticated methods (as applying either NCR or OSS once or even recursively reduces the size to the dataset only down to at most 120 000 samples, leaving the further reduction to 2000 samples to randomness), another option is to combine 1 and 2. With this in mind, we have devised an algorithm that we have termed *Recursive Targetted Removal* (RTR) and which adapts the chosen method to our problem by alternating between targetted and random removal. The steps are summarised in Algorithm 1.

Algorithm 1 Recursive Targetted Removal algorithm

```
1: procedure RTR(dataset)
2:     while size & balance unsatisfactory do
3:         while (size too large & size reduction relevant) or insufficient balance do
               targetted removal of majority class
4:         if size too large then randomly remove a percentage of negative instances
5:     return subsample
```

Note that the random removal is performed on the negative samples, in an attempt to make sure all positive samples are as informative as possible. Another option would be to alternate the random removal between the classes (something we discarded due to our preference for the positive class), but in any case it should be performed on only one class at a time, because its main function is to generate an imbalance for the targetted method to act. We refer to the corresponding notebooks for further details on the procedure.

To sum up, as we advanced in section 3, we implement two undersampling strategies aside from the baseline random selection: MTR and RTR, both with OSS and (alternatively) the NCR.

5. Spike-and-slab logistic regression

Introduction: Spike and Slab Priors

Spike-and-slab is a Bayesian variable selection technique [Tuchler(2008)] in which a random variable X either attains some fixed value v, called the *spike*, or is drawn some other prior $p_{slab}(x)$, called the *slab*. The usual way of constructing a spike and slab prior is to introduce a latent variable $Z \sim Ber(\theta)$ where $Z = 0$ means that X attains the fixed value v and Z = 1 means that X is drawn from the slab $p_{slab}(x)$:

$$Z \sim Ber(\theta), \tag{1}$$

$$X|Z = 0 \sim \delta(x - v), \tag{2}$$

$$X|Z = 1 \sim p_{slab}(x). \tag{3}$$

Marginalising over Z, we equivalently have that

$$X \sim \theta px(x) + (1 - \theta)\delta(x - v), \tag{4}$$

which we recognize as a mixture model with mixture components px_x and $\delta(x - v)$, respectively having weights θ and $1 - \theta$.

MCMC for Spike and Slab Regression

From the R package [Scott(2021)], we have selected a method called *logit.spike* that applies MCMC algorithm for logistic regression models with a *spike-and-slab* prior that places some amount of posterior probability at zero for a subset of the regression coefficients. The implementation can be found in `CERN-RandomSamples-Logistic-SpikeSlab.ipynb`, where we perform a random subsampling with 70000 samples belonging to each class and testing with the remaining samples.

6. Results

The table below shows the across-subsample average AUC and recall for the relevant cases considered: random undersampling with Gaussian prior (Random), MTR undersampling (with OSS and NCR), RTR undersampling (also with OSS and NCR) and random undersampling with a spike-and-slab prior (Random+SS):

Metric (%) \Model	Random	MTR (OSS)	MTR (NCR)	RTR (OSS)	RTR (NCR)	Random+SS
Train AUC	79.24	82.30	80.61	85.85	95.70	79.21
Test AUC	**78.95**	78.86	78.80	77.80	75.70	77.87
Train recall	74.30	78.30	76.83	80.87	93.23	74.25
Test recall	80.55	81.93	81.11	85.07	83.55	**87.96**

The bold figures indicate the best test values. In the cases where we have both JAGS and Stan implementations ("Random" and "RTR (OSS)") we have included only the JAGS result: we refer to the Stan notebooks for their evalutation. Additionally, in the "Random" case, we have not included the results for the horseshoe prior, as it provided no improvement in performance and we have already studied it in the second assignment (see `CERN-RandomSamples-Logistic-JAGS.ipynb` for the evaluation of the horseshoe implementation).

The first thing to note is that MTR gives very similar results to those of the baseline, with a moderate (yet valuable) improvement in recall. Such a small performance enhancement was expected, as most samples are still randomly selected.

Regarding the RTR implementations, they both show remarkable overfitting for AUC (the train value is larger than the test one), and RTR (NCR) also for recall (although less than for AUC). Additionally, both test AUCs turn out lower than the baseline's, while test recall is improved.

This result for the RTR is a little ambiguous: should we be happy about the better recall and forget about the AUC? We do not think so. As discussed in the corresponding notebooks, the variability across subsamples of the logistic parameters and of the recall suggests the RTR pipelines have an erratic behaviour. We find two possible reasons for this: either the logistic regression (being linear) is too simple a model to capture a higher-fidelity representation of a non-linear boundary, or the RTR representations are actually not capturing the real boundary, but *distorting* it. Indeed, we have discussed in section 4 that denoising strategies risk removing boundary-revealing points, and repeatedly alternating between targetted and random removal to force the desired final size could actually be *oversimplifying* the boundary instead of cleaning it. This second possibility would explain the overfitting: the resulting boundary is more easily modelled with by a logistic regression, but does not generalise; we would not have overfitting if the subsample boundary was too compicated/non-linear for the logistic regression to model[6].

Regarding the remarkably good test recall obtained with RTR, we believe it may be the consequence of a greater proportion of signal w.r.t. to the rest of cases (60% vs 50%; see the RTR notebooks). This is interesting, because it would suggest there is more

[6]In the case of RTR (OSS) one could, as we mention in `CERN-OssSamples-Logistic-JAGS.ipynb`, argue that the results owe to a smaller train size (1800 sanples in that case, versus the 2000 of the other undersampling schemes). However, as we explain in the notebook, we tried before with three greater subsamples (with size of approximately 2300) and we obtained slightly worse test results than the baseline's.

to imbalance problems than just seeking perfect balance: not only may there be an asymmetric distribution of useful information across classes in a given dataset (as we have mentioned), but also one may be interested in having a greater proportion of the signal of interest to enhance its identification beyond the reach of perfect balance.

As a matter of fact, that RTR oversimplifies the boundary is not so surprising given the magnitude of the dataset reduction we required: both one-step and recurrent targetted removal (especially the first) should give better results with a greater subsample size.

Turning now to the spike-and-slab, the test recall seems encouraging, but note that the train size used in that case is much greater (140 000 perfectly balanced samples); indeed, when using the same size as for the baseline (2000) the results are indistinguishable, so the spike-and-slab prior does not seem to contribute in this case (the same result as for the horseshoe prior).

Finally, let us briefly comment on absolute performance. The result is not bad if we consider that our train and test sizes for all cases but that of the spike-and-slab (2 000 vs 210 000), although the AUC obtained by the winners of the competition ("stack-exchangers") was 0.94 and our own best result (our team was called "Amanah") was as high as 0.90[7], both with a neural network (we however note that these were tested on a different dataset). The difference in performance, added to the RTR results, seems to indicate the dataset is not linearly separable.

7. Final remarks

In this project we have extended the treatment given in the second assignment with another unbalanced binary classification problem. We have implemented two new undersampling approaches based on the standard OSS and NCR methods: Minimal Targetted Removal and Recursive Targetted Removal. In addition, we have also tried the spike-and-slab shrinkage prior.

Our results show moderate improvement from the application of MTR, as expected, while RTR appears to oversimplify the decision boundary, leading to overfit, and the spike-and-slab prior bears no significant effect when applied to the same train size as the baseline (as happens also with the horseshoe prior).

Moreover, our absolute performance results are lower than the top results of the Kaggle leaderboard, which further points to the non-linearity of the decision boundary. This could have been dealt with by a non-linear model or the introduction of interaction variables, but since our goal was more to evaluate the undersampling approaches and the new prior introduced than to obtain the best possible performance, we refrained from such possibilities.

In any case, regardless of the final worth of the RTR method itself, its development and implementation in this project, as well as the introduction of the spike-and-slab prior, have provided what we consider valuable insight, namely as to the complexity of imbalance problems in general (as we have seen it is far from trivial to account for imbalance) and the potential use of inducing imbalance in favour of the class of interest.

[7]See https://www.kaggle.com/c/cernsignal/leaderboard.

Further development of the project could investigate the mentioned interaction features and more complex models to account for non-linearity (as it seems to place a hard upper bound on the performance of the logistic regression and thereby hinder to some extent the evaluation of the techniques implemented here), explore ways to implement cross-validation within our Bayesian paradigm (for a more robust model evaluation), check the performance of MTR and RTR with greater sample sizes (such as the one allowed by the spike-and-slab regression) or try other undersampling techniques.

References

[A. Dal Pozzolo(2015)] G. Bontempi A. Dal Pozzolo, O. Caelen. Package "unbalanced": Racing for unbalanced methods selection. *R package documentation*, 2015.

[Hart(1968)] P. Hart. The condensed nearest neighbor rule (corresp.). *IEEE Trans. Inf. Theory*, 14:515–516, 1968.

[Kubat and Matwin(1997)] Miroslav Kubat and Stan Matwin. Addressing the curse of imbalanced training sets: One-sided selection. In *In Proceedings of the Fourteenth International Conference on Machine Learning*, pages 179–186. Morgan Kaufmann, 1997.

[Laurikkala(2001)] Jorma Laurikkala. Improving identification of difficult small classes by balancing class distribution. pages 63–66, 06 2001. ISBN 978-3-540-42294-5. doi: 10.1007/3-540-48229-6_9.

[Scott(2021)] Steven L. Scott. Package 'boomspikeslab': Mcmc for spike and slab regression. *R package documentation*, 2021.

[Tomek(1976a)] I. Tomek. Two modifications of cnn. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-6(11):769–772, 1976a. doi: 10.1109/TSMC.1976.4309452.

[Tomek(1976b)] I. Tomek. An experiment with the edited nearest-neighbor rule. 1976b.

[Tuchler(2008)] Tuchler. Bayesian variable selection for logistic models using auxiliary mixture sampling. *Journal of Computational and Graphical Statistics*, 17:76 – 94, 2008.

[Wilson(1972)] Dennis L. Wilson. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-2(3): 408–421, 1972. doi: 10.1109/TSMC.1972.4309137.

[Zhang and Mani(2003)] J. Zhang and I. Mani. KNN Approach to Unbalanced Data Distributions: A Case Study Involving Information Extraction. In *Proceedings of the ICML'2003 Workshop on Learning from Imbalanced Datasets*, 2003.