



# EPSRC Centre for Doctoral Training in Industrially Focused Mathematical Modelling



## Herding top percentiles

Oliver Sheridan-Methven



## Contents

1.	Introduction .....	2
	Herding the top percentiles .....	2
	Glossary of terms .....	2
2.	Resampling .....	3
3.	Ensembles .....	4
	Adaptive boosting .....	4
	Gradient boosting .....	4
	Cost-sensitive boosting .....	4
4.	Experimental results .....	4
	Feature reduction .....	4
	Resampling .....	5
	Cost-sensitive boosting .....	5
	Final predictions .....	6
5.	Discussion, Conclusions & Recommendations .....	6
6.	Potential Impact .....	7



# 1. Introduction

A common problem occurring in many scientific, industrial, or governmental applications is *classification*. This is the process, known as *machine learning*, by which a machine is taught to learn patterns from a data-set and then make predictions on new data. The advantage of training machines to learn is the immense throughput they are capable of, and their ability to process highly complex data. Common examples include email classification, text recognition, and shopping recommendations.

Vodafone are a world leading mobile and communications provider, with over 444 million customers globally. These include 20 million UK mobile phone customers, and 30 million in Germany. Ensuring high customer satisfaction and continued customer service is a top priority for Vodafone. To achieve this, Vodafone look for patterns in their customers' behaviour, identify those who are dissatisfied with their phone contracts, and pre-empt cancellations with targeted promotions. With an abundance of data available, it is not clear what data are useful and what is not, nor what machine learning schemes are most suitable.

## Herding the top percentiles

Vodafone are interested in improving their *churn propensity model*. Usually when a customer is approaching the end of their contract, they will typically be looking to either renew it or terminate it. A customer is said to "*churn*" if they register an intent to cancel. Vodafone wish to pre-empt this, and if necessary provide that customer with a promotion before they churn. Typically, at any one time, approximately 8% of their customer base will churn within a 3-month period, and this increases to 15% when considering customers with between 5-8 months remaining on their contract. Hence, the customers they wish to identify and learn from only constitute a small minority of the data-set, producing an *imbalanced* classification problem.

Identifying customers who churn is a high priority to Vodafone

On top of having an imbalanced data-set to learn from, promotions are expensive, and Vodafone can only justify a few promotions, e.g. for 5-7% of their customers. Consequently, those most likely to churn and who are in the *top percentiles* when sorted by their probabilities of churning will be forwarded for promotions. It is only the classification accuracy in this percentile which is relevant. Increasing the proportion of churning customers in the upper percentiles is described as *herding*.

Our aim is to increase the classification accuracy in this top percentile, improving Vodafone's capabilities for predicting customer satisfaction.

## Glossary of terms

- **Algorithm:** An instruction set defining a numerical routine.
- **Learner:** A machine learning algorithm which can make predictions.
- **Churners:** Customers who register an intent to cancel their contract.
- **Herding:** Increasing the proportion of churners in the top percentiles.
- **Loss function:** Measures the degree of error in a learner's predictions.
- **Cross validation:** Repartitioning data multiple times to avoid over-fitting.
- **Feature:** A characteristic of a Vodafone customer which can be learnt.
- **Sample:** All the features of a single Vodafone customer.
- **Majority instance:** A sample corresponding to a satisfied customer.
- **Ensemble:** An aggregation of various learners, producing a single prediction.
- **Boosting:** An ensemble method, training learners in succession.

For machine learning, the data-set is split into training, validation, and prediction sets

Machine learning algorithms must be provided with data. The overall data-set is correspondingly split into three distinct and non-overlapping sets: *training*, *validation*, and *prediction*. The training set is what we provide to the algorithm to learn from. After a learner is trained, we assess its performance on the validation set. Validation identifies learners which may have over-fit the training set, or generalise poorly to unseen data. Depending on the results from the validation stage, we may revisit the algorithm, and iterate several times between training and validation. After this, we should know how the algorithms expect to perform, their robustness, and how they compare.

The marker we use to identify whether a customer should be offered a promotion is if they churned within 3 months. Hence, to create a predictive model, a learner must be trained on data which are at least 3 months old. The temporal structure of the data is presented in Figure 1. Each customer is a *sample* in this data-set. The samples from March 2017 form the prediction set. To keep the validation set representative of the prediction set, 400,000 samples were taken from December 2016 for validation. The training set consisted of the 397,028 samples remaining from 2016.

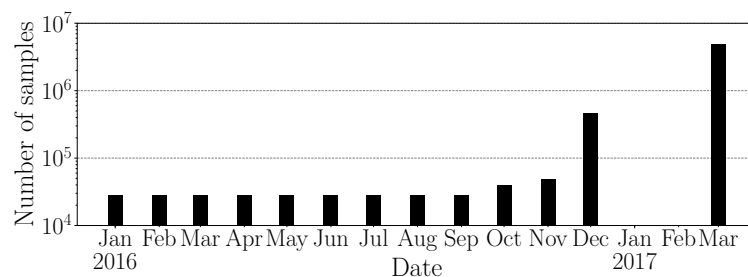


Figure 1: The temporal distribution of the samples.

The customers are only represented once in this data-set. Each sample has 211 features, which are mostly numeric, but contain some categorical and ordinal fields. Features include the number of calls, data usage, network coverage, province, etc. Naïvely, one could use all the features and expand the categorical variables, producing 291 features. However, if ordinals and categorical variables are omitted (along with some features Vodafone flagged as less useful), then only 134 features remain.

## 2. Resampling

A popular approach to tackle imbalanced data is to try and *rebalance* the data. This can be done by either undersampling the majority class (satisfied customers), or by oversampling the minority class (churners). Both have their pitfalls: undersampling may throw away important features in the training set, and oversampling may cause over-fitting.

A prominent algorithm for oversampling is the *synthetic minority oversampling technique* (SMOTE). This takes a minority class instance, finds the samples which resemble it most closely, and generates a new synthetic sample as an interpolation of these. A close relative of SMOTE is *adaptive synthetic sampling* (ADASYN), which samples frequently misclassified regions of the sample space which are deemed difficult to learn.

To undersample the majority class, one option is to randomly remove samples. A slightly more sophisticated approach is to try to remove instances which obscure boundaries between the classes. By removing these, the boundaries should become sharper and easier to learn. One method is to remove *Tomek links*. Two samples form a Tomek link if they are from different classes, and the majority instance is the closest resembling sample to the minority instance. If so, the majority instance is removed. This can be iterated several times, removing Tomek links on each parse. For the Vodafone data-set, approximately 10% of the majority instances were removed by two parses.

### 3. Ensembles

There are many algorithms we could use to learn from the data, each giving its own set of predictions. Furthermore, the way an algorithm learns may not be deterministic, and so the same algorithm may give different predictions over several training sessions. Multiple learners can be aggregated together to produce one overarching learner, called an *ensemble*. An ensemble should have a better overall accuracy (or stability) than its constituent *base learners*. We can create ensembles by *boosting*, which successively introduces new learners into the ensemble based on the performance of the most recently introduced base learner. We now present two boosting variants: *adaptive boosting* and *gradient boosting*.

#### Adaptive boosting

A very data-centric boosting method is adaptive boosting. The premise is to assume that several base learners have been trained and currently form an ensemble, but we want to introduce another. Taking the most recently trained learner, we see which samples from the training set it misclassifies. We then train a new learner on the same training set, but we reweight the samples. We increase the sample weights by a constant factor for those which were misclassified, and decrease those which were correctly classified. This means the new learner will increase its focus on learning those which were misclassified, and less on those which were correct. When this new learner is trained, the overall error it incurs is evaluated, and its contribution to the ensemble is weighted accordingly. The better the learner does, the greater its contribution to the ensemble.

Adaptive boosting increases the weights for misclassified samples

#### Gradient boosting

The performance of a learner can be quantified using a *loss function*, which measures a learner's degree of error. Traditional gradient boosting minimises this loss function using *steepest descent*, numerically computing how the loss function changes if the data is slightly altered. Learning this dependency, the boosting algorithm proposes an additional learner with different parameter values in the direction it expects to reduce the loss function quickest.

Extreme gradient boosting learns the Newton direction

An extension of steepest descent gradient boosting is *extreme gradient boosting*. This approximates the loss function around the existing learners, incorporating corrections for curvature. Minimising the loss function's increase, the resulting learner emulates the *Newton direction* of descent. An excellent implementation of extreme gradient boosting is XGBoost.

#### Cost-sensitive boosting

Both of these boosting methods assume the loss function is equally penalised for misclassified majority instances as for minority instances. However, recalling what the samples represent, it is much worse for Vodafone if a dissatisfied customer is overlooked than if a content customer is needlessly offered a promotion. Hence the cost of these two types of misclassification is not the same.

Different misclassification costs can be refactored into a *cost-sensitive* loss function. The regular adaptive boosting algorithm AdaBoost has such a cost-sensitive extension called AdaCost. There are many ways to incorporate cost-sensitivity; we construct two new algorithms: AdaCost-B1 which is a revised AdaCost with a different error measurement, and AdaCost-AATP which penalises misclassification in the top percentiles. Furthermore, we also construct a cost sensitive loss function for gradient boosting algorithms.

### 4. Experimental results

#### Feature reduction

The features are a mix of numeric, categorical, and ordinal values. If categorical and ordinal values are included, then there are 291 features, and only 134 if these are removed. Using XGBoost as a baseline method, we perform 15-fold cross validation. We find that the

accuracies in the top 5<sup>th</sup>-percentile are  $48.5 \pm 0.2\%$  using the 134 features, and  $47.1 \pm 0.4\%$  when including the ordinal and categorical variables. This implies we can use the reduced data-set, consistently achieve higher accuracies, and train algorithms faster than if we had used all the features.

## Resampling

To measure the efficacy of the various resampling methods, we record performances as an improvement (or reduction) relative to what XGBoost achieves without resampling. These improvements form a distribution, which we show in Figure 2. We immediately see ADASYN gives a substantial reduction in performance. Conversely, SMOTE gives an improvement of approximately 0.3%, although also produces an increased variance. Interestingly, removing Tomek links gives no nett improvement.

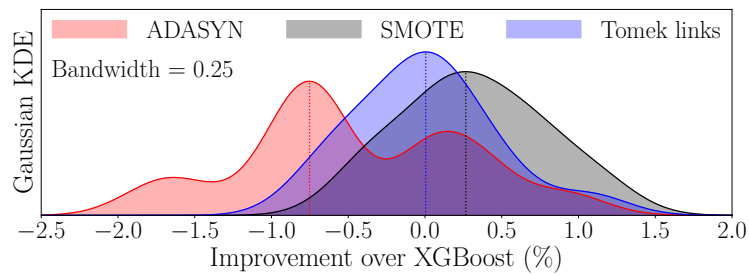


Figure 2: The performance gains from using various resampling methods with XGBoost.

## Cost-sensitive boosting

One of our results is that AdaCost appears to show improvements above AdaBoost. In light of this, we implement balanced (and imbalanced) ensembles using random undersampling. The resulting learners were combined using either AdaBoost or the AdaCost variants. The accuracies of these algorithms for different ensemble sizes are shown in Figure 3.

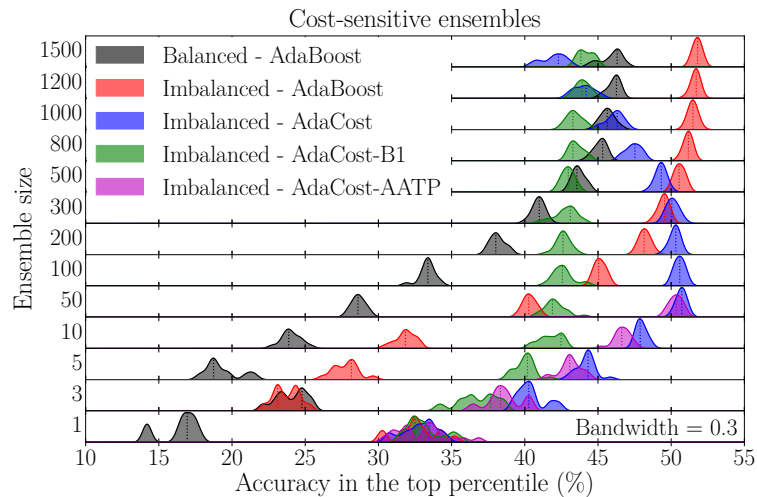


Figure 3: Accuracies in the top percentile using cost-sensitive ensembles.

Several results are embedded in Figure 3. We first highlight the improvement gained when AdaBoost uses imbalanced data. This learner achieves the best accuracy across all the ensembles and algorithms we tested, with peak performance at  $51.8 \pm 0.2\%$ , which is approximately 1.5% better than the best single algorithm Vodafone uses, and 1% better than the accuracy they achieve from a combination of 12 different algorithms.

Balancing the data reduced the accuracy in the top percentile

The cost-sensitive boosting methods exhibit very interesting scaling with increased ensemble sizes. The most significant is that AdaCost achieved a comparably high accuracy using a much smaller ensemble size of 50. However, for larger ensembles, this performance reverted. Frustratingly, while showing promise in earlier tests, AdaCost-B1 and AdaCost-AATP did not show a competitive edge.

We also explored cost-sensitive extreme gradient boosting, but were unable to demonstrate improved performances from introducing cost-sensitive loss functions. However, this was not exhaustive, and more work is required before dismissing this.

## Final predictions

After all the rounds of validation, we finally use the best performing algorithms on the final prediction set (containing 953,479 samples). We used XGBoost with and without SMOTE, and imbalanced AdaCost and AdaBoost for 50 and 1,500 base learners respectively. For comparison we also include *logistic regression*, (a simple case of linear models for binary classification). The accuracies achieved are show in Figure 4.

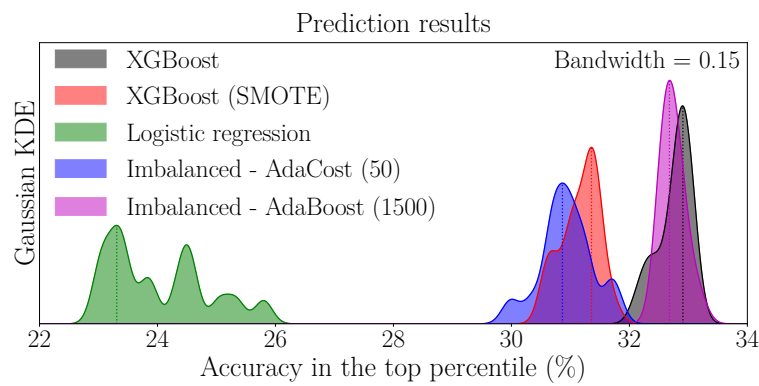


Figure 4: The final accuracies on the prediction set. Ensemble sizes are parenthesized.


Immediately we notice that these accuracies are all much lower than the 45-50% expected from the validation set results. However, given the temporal segregation of the data from Figure 1, it is not unreasonable to attribute this to an inherently more difficult data-set. We notice that XGBoost using SMOTE gives a worse performance than without, and AdaCost is appreciably worse than AdaBoost. However, AdaBoost with its imbalanced ensemble does achieve a competitive accuracy compared with XGBoost, which is as expected. Furthermore, we see AdaBoost shows a smaller variation than XGBoost.

## 5. Discussion, Conclusions & Recommendations

We have considered the problem of classifying churning Vodafone customers which is a percentile herding task with an underlying data imbalance. To tackle these challenges, we have investigated data processing and feature reduction, resampling methods, and boosting algorithms.

We inspected the data and demonstrated that higher accuracies could be achieved when using a reduced data-set. Selecting a subset of the features, ignoring the categorical and ordinal features, we reduced the number of features down from 291 to 134. The reduced set produced higher accuracies, more consistent results, and was faster and cheaper to compute.

Within the Vodafone customer base, there is an imbalance in the number of churners and non-churners. To address this imbalance, we considered ADASYN, SMOTE, and Tomek links, which were two oversampling methods and an undersampling method respectively. SMOTE showed small but appreciable performance gains, Tomek links gave no net



improvement, and ADASYN reduced performance. Consequently, we would recommend including SMOTE in subsequent algorithm validations.

AdaBoost achieved an accuracy of 51.8%

From the experiments with cost-sensitive boosting methods, we found a myriad of different performance behaviours. Emerging from this we found that AdaBoost, using 1,500 imbalanced base learners, achieved the highest recorded accuracy of  $51.8 \pm 0.2\%$ . This was approximately 1.5% better than the best single algorithm Vodafone had found.

As a consequence of these findings, we would promote exploring SMOTE when developing and validating any new algorithms. Furthermore, its benefits could be realised across a wider range of algorithms. Additionally, although cost-sensitive boosting algorithms did not achieve the highest recorded accuracy, for a variety of different base learners they usually proved competitive, and would be suitable for further comparisons.

## 6. Potential Impact

The challenges caused by imbalanced data, and the task of herding top percentiles, is a ubiquitous and pervasive problem frequently encountered across many areas of Vodafone's business. Having a better knowledge of approaches to tackle these difficulties, from processing the data to ensemble methods, presents many opportunities for an improved quality of service which can be passed onto Vodafone's customers.

Dr Uwe Bombosch, lead data scientist at Vodafone Germany, said *"Finding a top quantile with a high true positive rate is a ubiquitous problem in our business, in fact more wide-spread than a good overall classification. The outcome of the InFoMM mini-project can immediately be applied to several machine learning models we are building for marketing and customer satisfaction campaigns. SMOTE, for instance, has already found its way into a campaign channel optimisation. In addition to the insights, we also received a package of well-written code for our toolbox. When applying for a mini-project, I expected the highest level of maths, but was surprised by the IT-literacy and the hands-on approach. I clearly recommend InFoMM projects as a source of inspiration to data science organisations."*