Re-sampling Strategies for Regression

Luís Torgo[1,2], Paula Branco[1,2], Rita P. Ribeiro[1,2], and Bernhard Pfahringer[3]

[1]LIAAD - INESC TEC
[2] DCC - Faculdade de Ciências - Universidade do Porto
[3] Department of Computer Science - University of Waikato
ltorgo@dcc.fc.up.pt, paobranco@gmail.com, rpribeiro@dcc.fc.up.pt, bernhard@cs.waikato.ac.nz

February 22, 2016

Abstract

Several real world prediction problems involve forecasting rare values of a target variable. When this variable is nominal we have a problem of class imbalance which was thoroughly studied within machine learning. For regression tasks, where the target variable is continuous, few works exist addressing this type of problem. Still, important applications involve forecasting rare extreme values of a continuous target variable. This paper describes a contribution to this type of tasks. Namely, we propose to address such tasks by re-sampling approaches that change the distribution of the given data set to decrease the problem of imbalance between the rare target cases and the most frequent ones. We present two modifications of well-known re-sampling strategies for classification tasks: the under-sampling and the SMOTE methods. These modifications allow the use of these strategies on regression tasks where the goal is to forecast rare extreme values of the target variable. In an extensive set of experiments we provide empirical evidence for the superiority of our proposals for these particular regression tasks. The proposed re-sampling methods can be used with any existing regression algorithm, which means that they are general tools for addressing problems of forecasting rare extreme values of a continuous target variable.

1 Introduction

Forecasting rare extreme values of a continuous variable is very relevant for several real world domains (e.g. finance, ecology, meteorology, etc.). This problem can be seen as equivalent to classification problems with imbalanced class distributions which have been studied for a long time within

machine learning (e.g. Domingos (1999); Elkan (2001); Zadrozny (2005); Chawla (2005)). The main difference is the fact that we have a target numeric variable, i.e. a regression task. This type of problem is particularly difficult because: i) there are few examples with the rare target values; ii) the errors of the learned models are not equally relevant because the user's main goal is predictive accuracy on the rare values; and iii) standard prediction error metrics are not adequate to measure the quality of the models given the preference bias of the user.

The existing approaches for the classification scenario can be cast into 3 main groups (Zadrozny, 2003; Ling and Sheng, 2010): i) change the evaluation metrics to better capture the application bias; ii) change the learning systems to bias their optimization process to the goals of these domains; and iii) re-sampling approaches that manipulate the training data distribution so as to allow the use of standard learning systems. All these three approaches were extensively explored within the classification scenario (e.g. Kubat and Matwin (1997); Chawla et al. (2002)). Research work within the regression setting is much more limited. Torgo and Ribeiro (2009) and Ribeiro (2011) proposed a set of specific metrics for regression tasks with non-uniform costs and benefits. Ribeiro (2011) described system UBARULES that was specifically designed to address this type of problem. Still, to the best of our knowledge, no one has tried re-sampling approaches on this type of regression tasks. Nevertheless, re-sampling strategies have a clear advantage over the other alternatives - they allow the use of any existing regression tool on this type of tasks without the need to change it. The main goal of this paper is to explore this alternative within a regression context. We describe two possible methods: i) using an under-sampling strategy; and ii) using a SMOTE-like approach that performs both over- and under-sampling.

The main contributions of this work are: i) presenting a first attempt at addressing rare extreme values prediction using standard regression tools through re-sampling approaches; and ii) adapting two well-known and successful re-sampling methods for regression tasks. The results of the empirical evaluation of our contributions provide clear evidence on the validity of these approaches for the task of predicting rare extreme values of a numeric target variable. The significance of our contributions results from the fact that they allow the use of any existing regression tool on these important tasks by simply manipulating the available data set using our supplied code. All code and data used in this paper is provided in an associated web page[1] to ensure easy reproducibility and re-use of our results and algorithms.

[1] `http://www.dcc.fc.up.pt/~ltorgo/ExpertSystems`

2 Problem Formulation

Predicting rare extreme values of a continuous variable is a particular class of regression problems. In this context, given a training sample of the problem, $\mathcal{D} = \{\langle \mathbf{x}, y \rangle\}_{i=1}^{N}$, our goal is to obtain a model that approximates the unknown regression function $y = f(\mathbf{x})$. The particularity of our target tasks is that the goal is the predictive accuracy on a particular subset of the domain of the target variable Y - the rare and extreme values. As mentioned before, this is similar to classification problems with extremely unbalanced classes. As in these problems, the user goal is the performance of the models on a sub-range of the target variable values that is very infrequent. In this context, standard regression metrics (e.g. mean squared error) suffer from the same problems as error rate (or accuracy) on imbalanced classification tasks - they do not focus on the rare cases performance. In classification the solution usually revolves around the use of the precision/recall evaluation framework (Davis and Goadrich, 2006). Precision provides an indication on how accurate are the predictions of rare cases made by the model. Recall tells us how frequently the rare situations were signalled as such by the model. Both are important properties that frequently require some form of trade-off. How can we get similar evaluation for the numeric prediction of rare extreme values? On one hand we want that when our models predict an extreme value they are accurate (high precision), on the other hand we want our models to make extreme value predictions for the cases where the true value is an extreme (high recall). Assuming the user gives us information on what is considered an extreme for the domain at hand (e.g. $Y < k_1$ is an extreme low, and $Y > k_2$ is an extreme high), we could transform this into a classification problem and calculate the precision and recall of our models for each type of extreme. However, this would ignore the notion of numeric precision. Two predicted values very distant from each other, as long as being both extremes (above or below the given thresholds) would count as equally valuable predictions. This is clearly counter-intuitive on regression problems such as our tasks. A solution to this problem was described by Torgo and Ribeiro (2009) and Ribeiro (2011) that have presented a formulation of precision and recall for regression tasks that also considers the issue of numeric accuracy. We will use this framework to compare and evaluate our proposals for this type of tasks. For completeness, we will now briefly describe the framework proposed by Ribeiro (2011) that will be used in the experimental evaluation of our proposal[2].

[2]The code implementing this evaluation framework that was used in our experiments is available at `http://www.dcc.fc.up.pt/~rpribeiro/uba/`.

2.1 Utility-based Regression

The precision/recall evaluation framework we will use is based on the concept of utility-based regression (Ribeiro, 2011; Torgo and Ribeiro, 2007). At the core of utility-based regression is the notion of relevance of the target variable values and the assumption that this relevance is not uniform across the domain of this variable. This notion is motivated by the fact that contrary to standard regression, in some domains not all the values are equally important/relevant. In utility-based regression the usefulness of a prediction is a function of both the numeric error of the prediction (given by some loss function $L(\hat{y}, y)$) and the relevance (importance) of both the predicted $\hat{y}$ and true y values. Relevance is the crucial property that expresses the domain-specific biases concerning the different importance of the values. It is defined as a continuous function $\phi(Y) : \mathcal{Y} \rightarrow [0, 1]$ that maps the target variable domain $\mathcal{Y}$ into a $[0, 1]$ scale of relevance, where 0 represents the minimum and 1 represents the maximum relevance.

To better illustrate the concept of the relevance function, let us consider a regression problem where the goal is to predict the total percentage of burnt area on forest fires. Figure 1 shows a possible relevance function for this problem. It was created with the goal of stressing the domain-specific knowledge that large scale forest fires are the most important events we should aim at forecasting[3]. In effect, the accurate prediction of such type of fires has high benefits as it allows prevention and planning actions to be carried out and also avoids the large costs of not being prepared for these events.

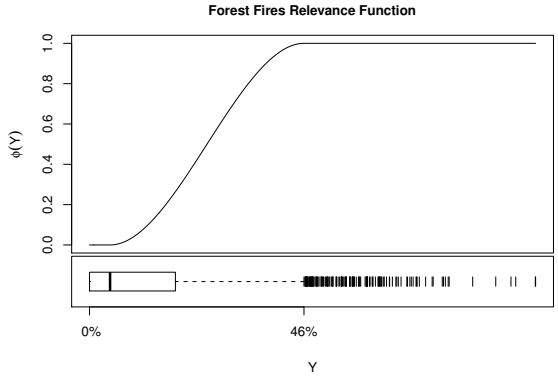

Figure 1: Relevance function ϕ for the percentage of burnt forest area.

According to Ribeiro (2011), the only assumption regarding the shape

[3]The function ϕ was obtained by applying *piecewise cubic Hermite interpolation* to the box-plot (Cleveland, 1993) statistics of the target variable as described in Ribeiro (2011). Still, any other method could have been used to define the relevance function.

of the function $\phi()$ is that there are some ranges of its domain where it assumes a quasi-concave shape, designated as *bumps of relevance* (see Ribeiro (2011) for more details). These *bumps* correspond to the ranges of the target variable values the user deems as more important. Being a domain-specific function, it is the user responsibility to specify the relevance function. However, Ribeiro (2011) describes some specific methods of obtaining automatically these functions when the goal is to be accurate at rare extreme values, which is the case of our applications. The methods are based on the simple observation that for these applications the notion of relevance is inversely proportional to the target variable probability density function. In our experiments we have used this strategy to come up with the relevance functions for the data sets we have used. The obtained relevance functions have essentially one[4] or two *bumps* of relevance associated with the high and low extreme values, and then a large region in the middle with near zero relevance, corresponding to the non-extreme and frequent values of the target variable.

The utility of a model prediction is related to the question on whether it has led to the identification of the correct type of extreme and if the prediction was precise enough in numeric terms. Thus to calculate the utility of a prediction it is necessary to consider two aspects: (i) does it identify the correct type of extreme? (ii) what is the numeric accuracy of the prediction (i.e. $L(\hat{y}, y)$)? This latter issue is important because it allows for coping with different "degrees" of actions as a result of the model predictions. For instance, in the context of financial trading an agent may use a decision rule that implies buying an asset if the predicted return is above a certain threshold. However, this same agent may invest different amounts depending on the predicted return, and thus the need for precise numeric forecasts of the returns on top of the correct identification of the type of extreme. This numeric precision, together with the fact that we may have more than one type of extreme (i.e. more than one "positive" class) are the key distinguishing features of this framework when compared to pure classification approaches, and are also the main reasons why it does not make sense to map our problems to classification tasks.

The concrete utility score of a prediction, in accordance with the original framework of utility-based learning (e.g. Elkan (2001); Zadrozny (2005)), results from the net balance between its benefits and costs (i.e. negative benefits). A prediction should be considered beneficial only if it leads to the identification of the correct type of extreme. However, the reward should also increase with the numeric accuracy of the prediction and should be dependent on the relevance of the true value. In this context, Ribeiro (2011) has defined the benefit of a prediction as a proportion of its maximum

[4]Some data sets only have one bump because they either only have high or low rare extreme values.

benefit that is given by the relevance of the true value, i.e. $\phi(y)$,

$$B_\phi(\hat{y}, y) = \phi(y) \cdot (1 - \Gamma_B(\hat{y}, y)) \tag{1}$$

where Γ_B is a *bounded loss function for benefits*.

The bounded loss Γ_B is a $[0, 1]$ function that maps the unbounded range of a standard regression loss function (e.g. squared error) into a proportion[5]. Γ_B is designed to give the value of 0 if we have a perfect prediction thus leading to the maximum benefits in the current situation, i.e. $\phi(y)$. The function smoothly increases up to 1 as we move away from a perfect prediction. Γ_B definition ensures that the benefits will be zero if either the prediction is too inaccurate or the predicted value is not a similar type of extreme as the true value.

Regarding costs, the rationale is that a prediction should entail a cost if it is too inaccurate and/or is unable to correctly identify the type of true value[6]. To quantify the cost of these erroneous predictions, it is necessary to determine how relevant the impact of such mistakes is. While the benefits of a prediction depend on the usefulness of its associated true value (i.e. $\phi(y)$), costs depend on both the relevance of the true and predicted values, because they are of different type. This means that costs are proportional to the relevance of both the true and predicted values. The *joint relevance function* captures this notion by calculating a weighted average of these two factors,

$$\phi^p(\hat{y}, y) = (1 - p) \cdot \phi(\hat{y}) \ + \ p \cdot \phi(y) \tag{2}$$

where $p \in [0, 1]$ is a factor differentiating the types of errors.

In the context of our target applications we can distinguish three different types of mistakes: (i) *false alarms* where we predict an extreme value for an irrelevant/normal true value ; (ii) *missed events* where the model predicts an irrelevant value but the true value is either an extreme high or low value; or (iii) *confusing events* where we predict a high(low) extreme for a true low(high) extreme value. This third scenario is the most serious type of mistake and also the one where the value of $\phi^p(\hat{y}, y)$ will be higher because both $\phi(\hat{y})$ and $\phi(y)$ will be high.

Ribeiro (2011) defined the cost of a prediction as a proportion of the maximum cost that is given by the joint relevance of both true and predicted values. Hence, the *cost function* C^p_ϕ is defined as,

$$C^p_\phi(\hat{y}, y) = \phi^p(\hat{y}, y) \cdot \Gamma_C(\hat{y}, y) \tag{3}$$

where ϕ^p is the joint relevance function and Γ_C is the bounded loss function for costs.

[5] See full details in Section 3.3 of Ribeiro (2011).

[6] Recall that we essentially have 3 types of values - extremes (high or low) or irrelevant values.

Similarly to Γ_B, the bounded loss Γ_C is a [0,1] function that determines the proportion of maximum cost assigned to a prediction, based on the prediction error measured by a standard loss function. It should be 0 for a perfect prediction and increase up to 1, as it moves away from a perfect prediction.

Given the above definitions of benefits and costs the utility associated to any prediction can be calculated as the net balance of these two factors,

$$\begin{aligned} U_\phi^p(\hat{y}, y) &= B_\phi(\hat{y}, y) \; - \; C_\phi^p(\hat{y}, y) \\ &= \phi(y) \cdot (1 - \Gamma_B(\hat{y}, y)) \; - \; \phi^p(\hat{y}, y) \cdot \Gamma_C(\hat{y}, y) \end{aligned} \tag{4}$$

where $B_\phi(\hat{y}, y)$, $C_\phi^p(\hat{y}, y)$, $\Gamma_B(\hat{y}, y)$ and $\Gamma_C(\hat{y}, y)$ are functions related to the notions of costs and benefits of predictions that are defined in Ribeiro (2011).

2.2 Precision and Recall for Regression

Precision and recall are two of the most commonly used metrics to estimate the performance of models in highly skewed domains (Davis and Goadrich, 2006) such as our target domains. The main advantage of these statistics is that they are focused on the performance on the target events, disregarding the remaining cases. In imbalanced classification problems, the target events are cases belonging to the minority (positive) class. Informally, precision measures the proportion of events signalled by the model that are real events, while recall measures the proportion of events occurring in the domain that are captured by the model.

The notions of precision and recall were adapted to regression problems with non-uniform relevance of the target values by Torgo and Ribeiro (2009) and Ribeiro (2011). In this paper we will use the framework proposed by these authors to evaluate and compare our sampling approaches. We will now briefly present the main details of this formulation[7].

The key issue of the precision/recall framework is the notion of interesting events. The definition of the cases which are to be considered interesting for the user is not so straightforward in regression as in classification. Ribeiro (2011) suggests the use of the relevance function to define the notion of interesting events for regression problems. Namely, a case is considered an interesting event if,

$$z := I(\phi(y) \geq t_E) \tag{5}$$

where I is the indicator function giving 1 if its argument is true and 0 otherwise, and $t_E \in [0, 1]$ is a domain-specific event threshold on the relevance scores.

[7] Full details can be obtained in Chapter 4 of Ribeiro (2011).

In order to calculate precision and recall it is also necessary to establish when a prediction is accurate concerning a value being or not an event. Intuitively, one could be tempted to consider a prediction as a correct forecast of the event if $\phi(\hat{y}) \geq t_E$. However, as mentioned by Ribeiro (2011), there might be cases where the relevance of the predicted value is high and nevertheless the prediction is incorrect. For instance, if the true target value is an extreme high value and the predicted value is an extreme low value, both will have high relevance score and still this is a serious error. The origin of the problem lies on the fact that contrary to the classification set-up, in utility-based regression there may exist more than one interesting "class", which correspond to more than one bump of relevance. This is a fundamental difference to the imbalanced classification framework.

The solution proposed by Ribeiro (2011) is to resort to the notion of utility of a prediction to decide its correctness. From the definition of utility (Equation 4) it follows that: (i) the maximum of $U^p_\phi(\hat{y}, y)$ is $\phi(y)$; and (ii) the minimum of $U^p_\phi(\hat{y}, y)$ is $-\phi^p(\hat{y}, y)$. In this sense, the higher the precision of the predictions the closer is U^p_ϕ to $\phi(y)$. This means that the utility score is strongly correlated with the probability that a prediction $\hat{y}$ is in the same bump as y, i.e. that it is a correct forecast of some event. Ribeiro (2011) proposes a method based on the calibration of the raw utility scores to reach the final decision on whether a predicted value is or not a correct "event" prediction, i.e. the value of $\hat{z}$. Namely, this binary property is defined as,

$$\hat{z} := I\left(s > \frac{1-u}{2}\right) \tag{6}$$

where s is a calibrated utility score, u is the raw utility score ($U^p_\phi(\hat{y}, y)$) and I is the indicator function giving 1 if its argument is true and 0 otherwise.

Precision and recall are usually defined as ratios between the correctly identified events (usually known as true positives within classification), and either the signalled events (for precision), or the true events (for recall). Given that the notion of event was defined based on the concept of utility, Ribeiro (2011) also defines the two denominators of these ratios as functions of utility, finally leading to the following definitions of precision and recall for regression,

$$recall = \frac{\sum\limits_{i:\hat{z}_i=1, z_i=1} (1 + u_i)}{\sum\limits_{i:z_i=1} (1 + \phi(y_i))} \tag{7}$$

and

$$precision = \frac{\sum_{i:\hat{z}_i=1,z_i=1} (1 + u_i)}{\sum_{i:\hat{z}_i=1,z_i=1} (1 + \phi(y_i)) + \sum_{i:\hat{z}_i=1,z_i=0} (2 - p\,(1 - \phi(y_i)))} \quad (8)$$

where p is a weight differentiating the types of errors, while $\hat{z}$ and z are binary properties associated with being in the presence of a rare extreme case.

Through the use of utility in our proposed definitions of precision and recall we are able to penalize not only *false alarms* (FP) and *missed opportunities* (FN), but also *confusing events*. In an application where both type of extremes, high and low, are of interest of the end user, a *confusing event* should be heavily penalized. In the stock market domain, for instance, if a model advises the user to buy when in fact should sell, it is a serious error. By using utility, that same prediction would be considered as a cost (i.e. negative utility value). Moreover, the amount of penalization is more sensitive, in the sense that it takes on a continuous scale depending on the difference between the true and predicted values because of the definition of utility.

In the experimental evaluation of our sampling approaches we have used as main evaluation metric the F-measure that can be calculated with the values of precision and recall,

$$F = \frac{(\beta^2 + 1) \cdot precision \cdot recall}{\beta^2 \cdot precision + recall} \quad (9)$$

where β is a parameter weighing the importance given to precision and recall (we have used $\beta = 1$, which means equal importance to both factors).

3 Re-sampling Approaches for Regression

The basic motivation for re-sampling approaches is the assumption that the imbalanced distribution of the given training sample will bias the learning systems towards solutions that are not in accordance with the user's preference goal. This occurs because the goal is predictive accuracy on the data that is least represented in the original data set. Most existing learning systems work by searching the space of possible models with the goal of optimizing some criteria. These criteria are usually related to some form of average performance. These metrics will tend to reflect the performance on the most frequent cases, which are not the goal of the user. In this context, the goal of re-sampling approaches is to change the data distribution on the training sample so as to make the learners focus on cases that are of interest

to the user. The change that is carried out has the goal of balancing the distribution of the least represented (but more important) cases with the more frequent observations.

Many re-sampling approaches exist within the imbalanced classification literature. To the best of our knowledge no attempt has been made to apply these strategies to the equivalent regression tasks - forecasting rare extreme values. In this section we describe the adaptation of two existing re-sampling approaches to these regression tasks.

3.1 Under-sampling common values

The basic idea of under-sampling (e.g. Kubat and Matwin (1997)) is to decrease the number of observations with the most common target variable values with the goal of better balancing the ratio between these observations and the ones with the interesting target values that are less frequent. Within classification this consists on obtaining a random sample from the training cases with the frequent (and less interesting) class values. This sample is then joined with the observations with the rare target class value to form the final training set that is used by the selected learning algorithm. This means that the training sample resulting from this approach will be smaller than the original (imbalanced) data set.

In regression we have a continuous target variable. As mentioned in Section 2.1 the notion of relevance can be used to specify the values of a continuous target variable that are more important for the user. We can also use the relevance function values to determine which are the observations with the common and uninteresting values that should be under-sampled. Namely, we propose the strategy of under-sampling observations whose target value has a relevance less than a user-defined parameter. This threshold will define the set of observations that are relevant according to the user preference bias,

$$\mathcal{D}_r = \{\langle \mathbf{x}, y \rangle \in \mathcal{D} : \phi(y) \geq t\} \tag{10}$$

where t is the user-defined threshold on relevance.

Under-sampling will be carried out on the remaining observations $\mathcal{D}_i = \mathcal{D} \setminus \mathcal{D}_r$. Regards the amount of under-sampling that is to be carried out the strategy is the following. For each of the relevant observations in $\mathcal{D}_r$ we will randomly select n_u cases from the "normal" observations in $\mathcal{D}_i$. The value of n_u is another user-defined parameter that will establish the desired ratio between "normal" and relevant observations. Too large values of n_u will result in a new training data set that is still too unbalanced, but too small values may result in a training set that is too small, particularly if there are too few relevant observations.

3.2 SMOTE for regression

SMOTE (Chawla et al., 2002) is a sampling method to address classification problems with imbalanced class distribution. The key feature of this method is that it combines under-sampling of the frequent classes with over-sampling of the minority class. Chawla et al. (2002) show the advantages of this approach when compared to other alternative sampling techniques on several real world problems using several classification algorithms. The key contribution of our work is to propose a variant of SMOTE for addressing regression tasks where the key goal is to accurately predict rare extreme values, which we will name SMOTER .

The original SMOTE algorithm uses an over-sampling strategy that consists on generating "synthetic" cases with a rare target value. Chawla et al. (2002) propose an interpolation strategy to create these artificial examples. For each case from the set of observations with rare values ($\mathcal{D}_r$), the strategy is to randomly select one of its k-nearest neighbours from this same set. With these two observations a new example is created whose attribute values are an interpolation of the values of the two original cases. Regards the target variable, as SMOTE is applied to classification problems with a single class of interest, all cases in $\mathcal{D}_r$ belong to this class and the same will happen to the new synthetic cases.

There are three key components of the SMOTE algorithm that we need to address in order to adapt it for our target regression tasks: i) how to define which are the relevant observations and the "normal" cases; ii) how to create new synthetic examples (i.e. over-sampling); and iii) how to decide the target variable value of these new synthetic examples. Regarding the first issue, the original algorithm is based on the information provided by the user concerning which class value is the target/rare class (usually known as the minority or positive class). In our problems we face a potentially infinite number of values of the target variable. As we have mentioned in Section 2.1, our proposal is based on the existence of a relevance function and on a user-specified threshold on the values of this function, that leads to the definition of the set $\mathcal{D}_r$ (c.f. Equation 10). Our algorithm will over-sample the observations in $\mathcal{D}_r$ and under-sample the remaining cases ($\mathcal{D}_i$), thus leading to a new training set with a more balanced distribution of the values. Regarding the second key component, the generation of new cases, we use the same approach as in the original algorithm though we have introduced some small modifications for being able to handle both numeric and nominal attributes. Finally, the third key issue is to decide the target variable value of the generated observations. In the original algorithm this is a trivial question, because as all rare cases have the same class (the target minority class), the same will happen to the examples generated from this set. In our case the answer is not so trivial. The cases that are to be over-sampled do not have the same target variable value, although they do have a

high relevance score ($\phi(y)$). This means that when a pair of examples is used to generate a new synthetic case, they will not have the same target variable value. Our proposal is to use a weighed average of the target variable values of the two seed examples. The weights are calculated as an inverse function of the distance of the generated case to each of the two seed examples.

Algorithm 1 The main SmoteR algorithm.

function SmoteR($\mathcal{D}, t_E, o, u, k$)
// $\mathcal{D}$ - A data set
// t_E - The threshold for relevance of the target variable values
// $\%o$,$\%u$ - Percentages of over- and under-sampling
// k - The number of neighbours used in case generation

$rareL \leftarrow \{\langle \mathbf{x}, y\rangle \in \mathcal{D} : \phi(y) > t_E \wedge y < \tilde{y}\}$ // $\tilde{y}$ is the median of the target Y
$newCasesL \leftarrow$ genSynthCases($rareL, \%o, k$) // generate cases for rareL
$rareH \leftarrow \{\langle \mathbf{x}, y\rangle \in \mathcal{D} : \phi(y) > t_E \wedge y > \tilde{y}\}$
$newCasesH \leftarrow$ genSynthCases($rareH, \%o, k$) // generate cases for rareH
$newRareCases \leftarrow rareL \bigcup newCasesL \bigcup rareH \bigcup newCasesH$
$tgtNorm \leftarrow \%u$ of $|newRareCases|$
$newNormCases \leftarrow$sample of $tgtNorm$ cases $\in \mathcal{D} \setminus \{rareL \bigcup rareH\}$
return $newRareCases \bigcup newNormCases$
end function

Algorithm 1 describes our proposed SmoteR sampling method. The algorithm uses a user-defined threshold (t_E) of relevance to define the sets $\mathcal{D}_r$ and $\mathcal{D}_i$. Notice that in our target applications we may have two rather different sets of rare cases: the extreme high and low values. This is another difference to the original algorithm. The consequence of this is that the generation of the synthetic examples is also done separately for these two sets. The reason is that although both sets include rare and interesting cases, they are of different type and thus with very different target variable values (extremely high and low values). The other parameters of the algorithm are the percentages of over- and under-sampling, and the number of neighbours to use in the cases generation. The key aspect of this algorithm is the generation of the synthetic cases (the two calls to function genSynthCases). This process is described in detail on Algorithm 2. The main differences to the original Smote algorithm are: the ability to handle both numeric and nominal variables; and the way the target value for the new cases is generated. Regards the former issue we simply perform a random selection between the values of the two seed cases. A possible alternative could be to use some biased sampling that considers the frequency of occurrence of each of the values within the rare cases. Regarding the target value we have used a weighted average between the values of the two seed cases. The weights are decided based on the distance between the new case and these two seed cases. The larger the distance the smaller the weight.

Algorithm 2 Generating synthetic cases.

function GENSYNTHCASES($\mathcal{D}, o, k$)

$newCases \leftarrow \{\}$
$ng \leftarrow \%o/100$ // nr. of new cases to generate for each existing case
for all $case \in \mathcal{D}$ **do**
  $nns \leftarrow \text{KNN}(k, case, \mathcal{D} \setminus \{case\})$ // k-Nearest Neighbours of $case$
  for $i \leftarrow 1$ **to** ng **do**
    $x \leftarrow$ randomly choose one of the nns
    for all $a \in$ attributes **do** // Generate attribute values
      if ISNUMERIC(a) **then**
        $diff \leftarrow case[a] - x[a]$
        $new[a] \leftarrow case[a] + \text{RANDOM}(0,1) \times diff$
      else
        $new[a] \leftarrow$ randomly select among $case[a]$ and $x[a]$
      end if
    end for
    $d_1 \leftarrow \text{DIST}(new, case)$ // Decide the target value
    $d_2 \leftarrow \text{DIST}(new, x)$
    $new[Target] \leftarrow \frac{d_2 \times case[Target] + d_1 \times x[Target]}{d_1 + d_2}$
    $newCases \leftarrow newCases \bigcup \{new\}$
  end for
end for
return $newCases$
end function

4 Experimental Evaluation

The goal of our experiments is to test the effectiveness of our proposed sampling approaches at predicting rare extreme values of a continuous target variable. For this purpose we have selected 17 regression data sets whose main characteristics are described in Table 1. For each of these data sets we have obtained a relevance function using the automatic method proposed by Ribeiro (2011). This method assigns higher relevance for values above (below) the adjacent values of the target variable sample distribution. These are calculated as a function of the quartiles and the inter-quartile range and are well-known thresholds for considering a value as an outlier. The result of applying this method are relevance functions that assign higher relevance to high and low rare extreme values, which are the target of the work in this paper. Based on these relevance functions we have imposed a threshold of 0.7 on the values of $\phi(Y)$ as the condition for a value to be taken as a rare extreme. As it can be seen from the information in Table 1 this leads to an average of 10-15% of the available cases having a rare extreme value for most data sets considered in our experiments.

In order to avoid any algorithm-dependent bias distorting our experimental results, we have carried out our comparisons using a diverse set of standard regression algorithms. Moreover, for each algorithm we have

Data Set	N	p	$nRare$	$\%Rare$	Data Set	N	p	$nRare$	$\%Rare$
a1	198	12	32	0.162	dAiler	7129	6	617	0.087
a2	198	12	26	0.131	availPwr	1802	16	197	0.109
a3	198	12	34	0.172	bank8FM	4499	9	384	0.085
a4	198	12	37	0.187	cpuSm	8192	13	804	0.098
a5	198	12	22	0.111	dElev	9517	7	1109	0.117
a6	198	12	35	0.177	fuelCons	1764	38	225	0.128
a7	198	12	29	0.146	boston	506	14	74	0.146
Abalone	4177	9	882	0.211	maxTorque	1802	33	163	0.09
Accel	1732	15	106	0.061					

Table 1: Used data sets and characteristics (N: n. of cases; p: nr. of predictors; $nRare$: nr. cases with $\phi(Y) > 0.70$; $\%Rare$: $nRare/N$).

Learner	Parameter Variants	R package
MARS	$nk = \{10, 17\}, degree = \{1, 2\}, thresh = \{0.01, 0.001\}$	**earth** (Milborrow, 2012)
SVM	$cost = \{10, 150, 300\}, gamma = \{0.01, 0.001\}$	**e1071** (Dimitriadou et al., 2011)
Random Forest	$mtry = \{5, 7\}, ntree = \{500, 750, 1500\}$	**randomForest** (Liaw and Wiener, 2002)

Table 2: Regression algorithms and parameter variants, and the respective R packages.

considered several parameter variants. Table 2 summarizes the learning algorithms that were used and also the respective parameter variants. For instance, for the MARS regression algorithm we have considered the 8 variants resulting from all combinations of parameter values reported in Table 2. To ensure easy replication of our work we have used the implementations of these algorithms available in the free and open source R environment (R Core Team, 2013), which is also the infrastructure used to implement our proposed sampling methods.

Each of the 20 learning approaches (8 MARS variants + 6 SVM variants + 6 Random Forest variants), were applied to each of the 17 regression problems using 13 different sampling approaches, namely: i) carrying out no sampling at all (i.e. use the data set with the original imbalance); ii) 8 variants of our SmoteR method; and iii) 4 variants of under-sampling. The 8 SmoteR variants used 5 nearest neighbours for case generation, a relevance threshold of 0.70 and all combinations of $\{200, 500\}\%$ and $\{50, 100, 200, 300\}\%$ for percentages of over- and under-sampling, respectively (c.f. Algorithm 1). For instance, in the data set *availPwr*, which contains 197 rare cases (c.f. Table 1), when using 200% oversampling and 50% under-sampling, we would generate 2 new cases for each existing rare case thus leading to 591 ($= 197 + 197 \times 2$) rare cases and 295 normal cases (50% of 591), thus a training set of 886 cases, whilst the original data set contains 1802 cases. With respect to under-sampling the 4 variants used $\{50, 100, 200, 300\}\%$ as percentages of under-sampling and the same 0.70

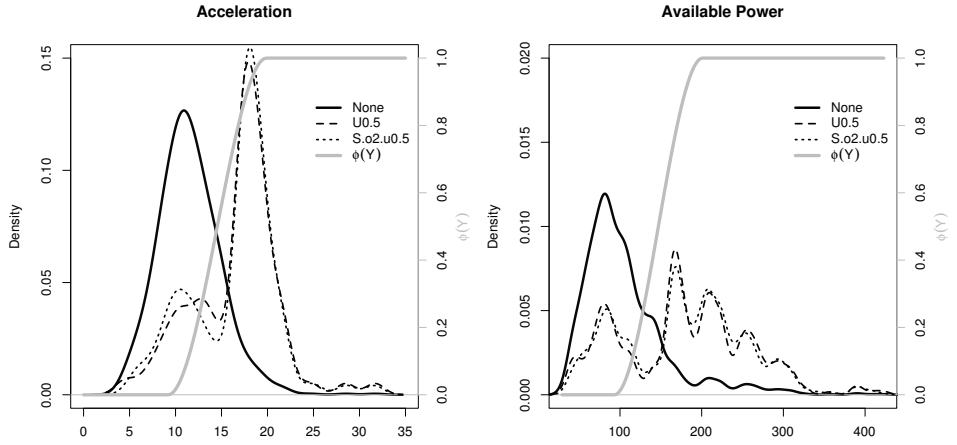

Figure 2: Distribution of the target variable before and after re-sampling.

relevance threshold. This means that if applying under-sampling with 50% to the same *availPwr* data set, we would end up with a training set formed by 197 rare cases plus 98 (50 % of 197) normal cases.

To have a better idea on the impact of these re-sampling strategies on the training set that is finally given to the learners, Figure 2 shows the distribution of the target variable[8] on the original data and on the data sets resulting from applying two of the most successful variants of our re-sampling strategies, for 2 specific data sets. The graphs on this figure clearly illustrate the change in the target variable distribution that is carried out by these re-sampling strategies with the goal of biasing this distribution towards the areas where the relevance function has higher values. Moreover, as illustrated above the methods also change the number of cases used for training, which will obviously have an impact on the computation time taken to obtain the models. More specifically for the data sets in Figure 2, the original *Acceleration* data set contains 1732 observations and the *S.o2.u0.5* configuration of SmoteR leads to a training set of 477, whilst the *U0.5* under-sampling variant uses only 159 cases. With respect to the *Available Power* data set the original size is 1802 and the two re-sampling variants use 886 and 295, respectively. Given that the SmoteR variant uses a more complex algorithm to obtain the training set involving distance calculations to find the nearest neighbours of rare cases, and leads to a larger training set, we can immediately see that this re-sampling approach requires more computation time than the simpler under-sampling strategy. Still, under these two particular configurations the training set will always be smaller than the original data set.

The goal of our experiments is to compare the 12 (8 SmoteR + 4 under-

[8]Approximated through a kernel density estimator.

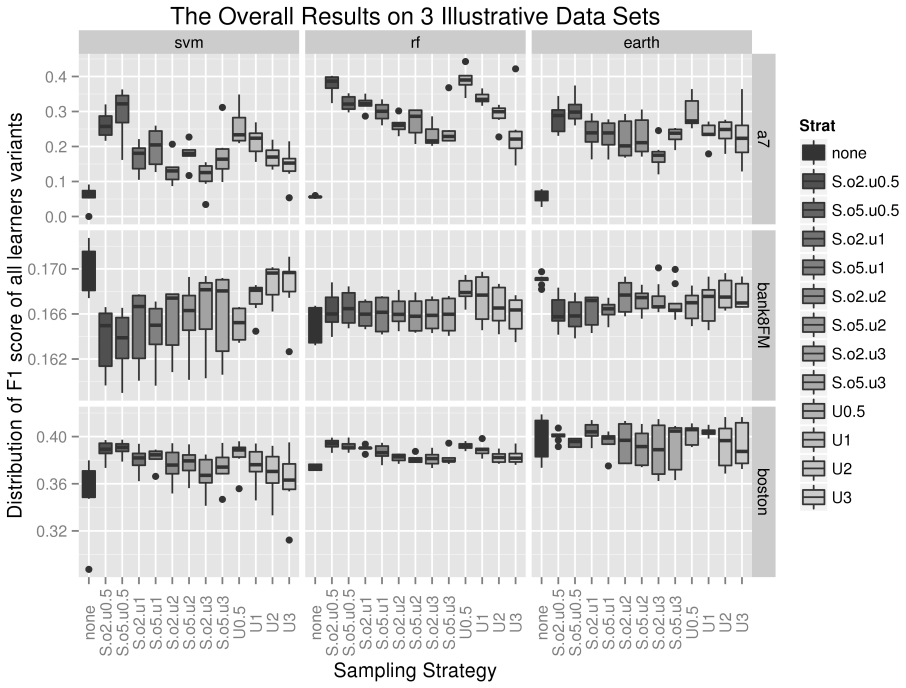

Figure 3: Behaviour of the sampling strategies on 3 illustrative data sets (S - SmoteR ; U - under-sampling; ox - $x \times 100\%$ over-sampling; ux - $x \times 100\%$ under-sampling).

sampling) re-sampling approaches against the default of learning with the given data, using 20 learning approaches on 17 data sets. All alternatives were evaluated according to the F-measure with $\beta = 1$ (c.f. Equation 2.2), which means that the same importance was given to both precision and recall scores that were calculated using the set-up described in Section 2.2. The values of the F-measure were estimated by means of 3 repetitions of a 10-fold cross validation process and the statistical significance of the observed paired differences was measured using the non-parametric Wilcoxon paired test.

Given the amount of experimental variants it is not possible to show all results given the space restrictions of a paper. Figure 3 shows the results for 3 of our 17 data sets. The full results for all data sets can be seen in the web page associated with the paper. For each combination of data set and regression algorithm the graph provides 13 box-plots, one for each of the 12 mentioned sampling approaches plus the alternative of using the original data (tagged as *none* in the graphs). The box plots show the distribution of the F1 scores of all variants of each learner on each data set. All scores are estimated by means of a $3 \times 10-$fold cross validation process. These

Sampling Strat.	Win (99%)	Win (95%)	Loss (99%)	Loss (95%)	Insignif. Diff.
U0.5	243	31	17	5	44
U1	213	41	10	4	72
U2	184	36	7	2	111
U3	138	42	11	3	146
S.o2.u0.5	229	31	14	6	60
S.o5.u0.5	216	42	15	3	64
S.o2.u1	201	52	12	3	72
S.o5.u1	199	44	14	3	80
S.o2.u2	173	43	11	2	111
S.o5.u2	172	46	13	1	108
S.o2.u3	127	69	17	2	125
S.o5.u3	142	46	15	6	131

Table 3: Summary of the paired comparisons to the no sampling baseline.

3 particular data sets were chosen because they represent three different patterns of results. Results on data set *a7* are among the best from the re-sampling approaches perspective. The *boston* data set can be regarded as an example of a domain where the advantage of re-sampling approaches is not so marked, while data set *bank8FM* is an example where re-sampling approaches behave poorly with the exception of random forest variants.

When taking into consideration all 17 data sets, in most cases we have an advantage of the re-sampling approaches. This can be confirmed when looking at the overall results in terms of the statistical significance of the paired differences between each sampling approach and the alternative of using the original data (the baseline). Table 3 summarizes the results of these paired comparisons. Each sampling strategy was compared against the baseline 340 times (20 learning variants times 17 data sets). For each paired comparison we check the statistical significance of the difference between the average F score obtained with the respective sampling approach and with the baseline. These averages were estimated using a 3×10-fold CV process. We counted the number of statistically significant wins and losses of each of the 12 sampling variants on these 340 paired comparisons using two significance levels (99% and 95%).

The results of Table 3 provide clear evidence of the advantage of using re-sampling approaches when the task is to predict rare extreme values of a continuous target variable. In effect, we can observe an overwhelming advantage in terms of number of statistically significant wins over the alternative of using the data set as given (i.e. no re-sampling). For instance, the particular configuration of using under-sampling with 50% (*U0.5*) was significantly better than the alternative of using the given data set on 80.5% ((243 + 31)/340) of the 340 considered situations, while only on 6.4% of the cases under-sampling actually lead to a significantly worse model. The

remarkable performance of this very simple re-sampling strategy is even re-enforced by the fact that it uses a much smaller training set than the other alternatives, which means lower computation costs. The SmoteR variant with 200% over-sampling and 50% under-sampling (*S.o2.u0.5*) also achieved very good results (76.5% significant wins). A pattern of results we have observed in our experiments is that higher values of the percentage of under-sampling leads to worse results. The lower this percentage the higher the imbalance in favour of the rare cases (for instance with 50% under-sampling there will be twice as many rare cases as normal cases). This has improved performance in all our experimental set ups. Another conclusion from our experiments is that increasing the percentage of over-sampling (i.e. generating more synthetic cases) makes the results worse. This may be an indication that the process being used for generating new cases may be taking too many risks leading to a degradation of the accuracy of the models.

In summary, the results of our experimental comparisons provide clear evidence on the validity of the re-sampling approaches we have considered, with particularly good results being obtained with training sets that bias the distribution towards a higher frequency of rare cases (i.e. do strong under-sampling of the normal cases).

5 Conclusions

This paper has presented a general approach to tackle the problem of forecasting rare extreme values of a continuous target variable using standard regression tools. The key advantage of the described re-sampling approaches is their simplicity. They allow the use of standard out-of-the-box regression tools on these particular prediction tasks by simply manipulating the distribution of the available training data.

The key contributions of this paper are : i) showing that re-sampling approaches can be successfully applied to this type of regression tasks; and ii) adapting two of the most successful sampling methods (Smote and under-sampling) to regression tasks.

The large set of experiments we have carried out on a diverse set of problems and using rather different learning algorithms, highlights the advantages of our proposals when compared to the alternative of simply applying the algorithms to the available data sets. Particularly noticeable are the results obtained with strong under-sampling of the normal cases. Moreover, in the case of under-sampling with 50% the good predictive performance is accompanied by lower computation costs.

R code implementing both the SmoteR and under-sampling strategies described in Section 3 is freely provided at `http://www.dcc.fc.up.pt/~ltorgo/ExpertSystems`. This URL also includes all code and data sets

necessary to replicate the experiments in the paper as well as extra experimental results that we could not fit within the paper.

Acknowledgements

This work is part-funded by the ERDF - European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness), by the Portuguese Funds through the FCT (Portuguese Foundation for Science and Technology) within project FCOMP - 01-0124-FEDER-022701.

References

Chawla, N. V. (2005). Data mining for imbalanced datasets: An overview. In *The Data Mining and Knowledge Discovery Handbook*. Springer.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: Synthetic minority over-sampling technique. *JAIR*, 16:321–357.

Cleveland, W. (1993). *Visualizing Data*. Hobart Press.

Davis, J. and Goadrich, M. (2006). The relationship between precision-recall and roc curves. In *ICML'06: Proc. of the 23rd Int. Conf. on Machine Learning*, ACM ICPS, pages 233–240. ACM.

Dimitriadou, E., Hornik, K., Leisch, F., Meyer, D., and Weingessel, A. (2011). *e1071: Misc Functions of the Department of Statistics (e1071), TU Wien*.

Domingos, P. (1999). Metacost: A general method for making classifiers cost-sensitive. In *KDD'99: Proceedings of the 5th International Conference on Knowledge Discovery and Data Mining*, pages 155–164. ACM Press.

Elkan, C. (2001). The foundations of cost-sensitive learning. In *IJCAI'01: Proc. of 17th Int. Joint Conf. of Artificial Intelligence*, volume 1, pages 973–978. Morgan Kaufmann Publishers.

Kubat, M. and Matwin, S. (1997). Addressing the curse of imbalanced training sets: One-sided selection. In *Proc. of the 14th Int. Conf. on Machine Learning*, pages 179–186. Morgan Kaufmann.

Liaw, A. and Wiener, M. (2002). Classification and regression by randomforest. *R News*, 2(3):18–22.

Ling, C. and Sheng, V. (2010). Cost-sensitive learning and the class imbalance problem. In *Encyclopedia of Machine Learning*. Springer.

Milborrow, S. (2012). *earth: Multivariate Adaptive Regression Spline Models. Derived from mda:mars by Trevor Hastie and Rob Tibshirani.*

R Core Team (2013). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Ribeiro, R. P. (2011). *Utility-based Regression*. PhD thesis, Dep. Computer Science, Faculty of Sciences - University of Porto.

Torgo, L. and Ribeiro, R. P. (2007). Utility-based regression. In *PKDD'07: Proc. of 11th European Conf. on Principles and Practice of Knowledge Discovery in Databases*, pages 597–604. Springer.

Torgo, L. and Ribeiro, R. P. (2009). Precision and recall in regression. In *DS'09: 12th Int. Conf. on Discovery Science*, pages 332–346. Springer.

Zadrozny, B. (2003). *Policy mining: Learning decision policies from fixed sets of data*. PhD thesis, University of California, San Diego.

Zadrozny, B. (2005). One-benefit learning: cost-sensitive learning with restricted cost information. In *UBDM'05: Proc. of the 1st Int. Workshop on Utility-Based Data Mining*, pages 53–58. ACM Press.