Protected probabilistic classification

Vladimir Vovk, Ivan Petej, and Alex Gammerman

практические выводы
теории вероятностей
могут быть обоснованы
в качестве следствий
гипотез о *предельной*
при данных ограничениях
сложности изучаемых явлений

On-line Compression Modelling Project (New Series)

Working Paper #35

First posted July 4, 2021. Last revised October 22, 2021.

Project web site:
http://alrw.net

Abstract

This paper proposes a way of protecting probabilistic prediction models against changes in the data distribution, concentrating on the case of classification and paying particular attention to binary classification. This is important in applications of machine learning, where the quality of a trained prediction algorithm may drop significantly in the process of its exploitation. Our techniques are based on recent work on conformal test martingales and older work on prediction with expert advice, namely tracking the best expert.

Contents

1 Introduction

A common problem in applications of machine learning is that, soon after a prediction algorithm is trained, the distribution of the data may change, and the prediction algorithm may need to be retrained. There are efficient ways of online detection of a change in distribution, such as using conformal test martingales [23], but there are inevitably awkward gaps between the change in distribution and its detection and between the detection of the change and the deployment of a retrained prediction algorithm.

We will refer to the trained prediction algorithm as our *prediction model*. This paper proposes a way of preventing a catastrophic drop in the quality of the prediction model when the data distribution changes. Given a prediction model, our procedure gives a protected prediction model that is more robust to changes in the data distribution. To use Anscombe's [1] insurance metaphor (repeatedly used already in [11]), our procedure provides an insurance policy against such changes. The main desiderata for such a policy are its low price and its efficiency.

The case of regression was discussed in an earlier paper [24]. In this paper we concentrate on the simpler case of classification, and first of all binary classification. We will often assume that the label space is $\{0, 1\}$. Suppose we are given a predictive system that maps past data and an object x with an unknown label $y \in \{0, 1\}$ to a number $p \in [0, 1]$, interpreted as the predicted probability that $y = 1$. We will refer to it as our *base predictive system*. In this paper we are mostly interested in the case where the base predictive system is a prediction model obtained by training a prediction algorithm, and so the predicted probability that $y = 1$ depends only on the object x, but allowing the dependence on the past data does not complicate the exposition.

We will be interested in two seemingly different questions about the base predictive system:

Online testing Can we gamble successfully against the base predictive system (at the odds determined by its predicted probabilities)? We are interested in online testing [23], i.e., in constructing test martingales (nonnegative martingales with initial value 1) with respect to the base predictive system that take large values on the actual sequence of observations.

Online prediction Can we improve the base predictive system, modifying its predictions p_n to better predictions p'_n?

If the quality of online prediction is measured using the log-loss function [8], the difference between the two questions almost disappears, as we will see in Sections 3 and 5.

After discussing online testing in Section 2 and online prediction in Section 3, we will give an example of a theoretical performance guarantee for our prediction procedure (an application of a known technique) in Section 4. In Section 5 we report encouraging experimental results, and Section 6 concludes.

Two appendixes contain proofs and further experimental results. Our computer code for the experiments is released in the form of Jupyter notebooks.

Algorithm 1 Simple Jumper martingale $((p_1, p_2, \dots) \mapsto (S_1, S_2, \dots))$

1: $C_\theta := 1_{\theta=\mathbf{0}}$ for all $\theta \in \Theta$
2: $C := 1$
3: **for** $n = 1, 2, \dots$:
4: **for** $\theta \in \Theta$: $C_\theta := (1 - J)C_\theta + (J/|\Theta|)C$
5: **for** $\theta \in \Theta$: $C_\theta := C_\theta B_{f_\theta(p_n)}(\{y_n\})/B_{p_n}(\{y_n\})$
6: $S_n := C := \sum_{\theta \in \Theta} C_\theta$

2 Testing predictions by betting

We consider a potentially infinite sequence of *actual observations* $z_1, z_2, \dots$, each consisting of two components: $z_n = (x_n, y_n)$, where $x_n \in \mathbf{X}$ is an *object* chosen from an *object space* $\mathbf{X}$, and $y_n \in \{0, 1\}$ is a binary label. A *predictive system* is a function that maps any object x and any finite sequence of observations $z_1, \dots, z_i$ (intuitively, the past data) for any $i \in \{0, 1, \dots\}$ to a number $p \in [0, 1]$ (intuitively, the probability that the label of x is 1). Fix a *base predictive system*, and let $p_1, p_2, \dots$ be its predictions for the actual observations: p_n is the prediction output by the base predictive system on x_n and $z_1, \dots, z_{n-1}$; it is interpreted as the predicted probability that $y_n = 1$. (We will not need any measurability assumptions; in particular, $\mathbf{X}$ is not supposed to be a measurable space.)

In this paper we are mostly interested in the special case where the output p of the base predictive system depends only on x and not on $z_1, \dots, z_i$. In this case we will say that our predictive system is a *prediction model*. A typical way in which prediction models appear in machine learning is as result of training a prediction algorithm. In Section 5 we will only consider prediction models, but for now we do not impose this restriction.

Our first online testing procedure is given as Algorithm 1, where we use the notation 1_E to mean 1 if a property E holds and 0 if not. One of its two parameters is a finite non-empty family $f_\theta : [0, 1] \to [0, 1]$, $\theta \in \Theta$, of *calibrating functions*. The intuition behind f_θ is that we are trying to improve the base predictions p_n, or *calibrate* them; the idea is to use a new prediction $f_\theta(p_n)$ instead of p_n. We assume that Θ contains a distinguished element $\mathbf{0} \in \Theta$ (used in line 1 of Algorithm 1).

Perhaps the simplest choice (explored in Appendix B) is to use a finite subset of the family

$$f_\theta(p) := p + \theta p(1 - p), \tag{1}$$

where $\theta \in [-1, 1]$, and $\theta = 0$ is the distinguished element. For $\theta > 0$ we are correcting for the forecasts p being underestimates of the true probability of 1, while for $\theta < 0$ we are correcting for p being overestimates; f_0 is the identity function (no correction).

We do not know in advance which f_θ will work best, and moreover, it seems plausible that suitable values of θ will change over time. Therefore, we use the idea of "tracking the best expert" [10]. Algorithm 1 uses the notation B_p,

$p \in [0,1]$, for the Bernoulli distribution on $\{0,1\}$ with parameter p: $B_p(\{1\}) = p$. To each sequence $\theta = (\theta_1, \theta_2, \dots)$ of elements of Θ corresponds the *elementary test martingale*

$$\prod_{i=1}^{n} \frac{B_{f_{\theta_i}(p_i)}(\{y_i\})}{B_{p_i}(\{y_i\})}, \quad n = 0, 1, \dots. \tag{2}$$

The other parameter of the Simple Jumper martingale of Algorithm 1 is $J \in [0,1]$, the *jumping rate*. This martingale is obtained by "derandomizing" (to use the terminology of [20]) the stochastic test martingale corresponding to the probability measure μ on $(\theta_1, \theta_2, \dots) \in [0,1]^\infty$ defined as the probability distribution of the following Markov chain with state space Θ. The initial state θ_0 (ignored by μ) is $\mathbf{0}$ (line 1 of Algorithm 1), and the transition function prescribes maintaining the same state with probability $1 - J$ and, with probability J, choosing a new state from the uniform probability measure on Θ (line 4). We will sometimes refer to it as the *Simple Jumper Markov chain*.

We derandomize the stochastic test martingale by averaging,

$$S_n := \int \prod_{i=1}^{n} \frac{B_{f_{\theta_i}(p_i)}(\{y_i\})}{B_{p_i}(\{y_i\})} \mu(\mathrm{d}\theta),$$

which gives us a deterministic test martingale. This construction is standard in conformal testing [23, Section 3].

In one respect the Simple Jumper martingale is not adaptive enough: we assume that a suitable jumping rate J is known in advance. Instead, we will use the *Composite Jumper* martingale that depends on two parameters, $\pi \in (0,1)$ and a finite set $\mathbf{J}$ of non-zero jumping rates. It is defined to be the weighted average

$$S_n := \pi + \frac{1-\pi}{|\mathbf{J}|} \sum_{J \in \mathbf{J}} S_n^J, \tag{3}$$

where S^J is computed by Algorithm 1 fed with J as its parameter.

In Appendix B we will see that already the simple choice (1) leads to very successful betting for popular benchmark datasets. However, there are numerous other natural calibration functions, some of which are shown in Figure 1. The function in red is in the quadratic family (1) (and its parameter is $\theta = 1$); these functions are fully above, fully below, or (for $\theta = 0$) situated on the bisector of the first quadrant (shown as the dotted line). In many situations other calibration functions will be more suitable. For example, it is well known that untrained humans tend to be overconfident [12, Part VI, especially Chapter 22]. A possible calibration function correcting for overconfidence is the cubic function

$$f_{a,b}(p) := p + ap(p-b)(p-1), \tag{4}$$

where $(a,b) = \theta \in [0,1]^2$. An example of such a function is shown in Figure 1 in blue (with parameters $a = 1.5$ and $b = 0.5$). The meaning of the parameters is that b is the value of p (such as 0.5) that we believe does not need correction, and that a indicates how aggressively we want to correct for overconfidence ($a < 0$

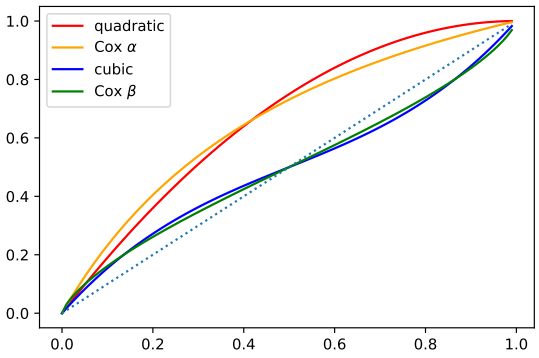

Figure 1: Examples of calibration functions.

meaning that in fact we are correcting for underconfidence). If the predictor predicts a p that is close to 0 or 1, we correct for his overconfidence (assuming $a > 0$) by moving p towards the neutral value b.

In the experimental Section 5 we will use Cox's [3, Section 3] calibration functions

$$f_{\alpha,\beta}(p) := \frac{p^\beta \exp(\alpha)}{p^\beta \exp(\alpha) + (1-p)^\beta}, \tag{5}$$

where $(\alpha, \beta) = \theta \in \mathbb{R}^2$. We obtain the identity function $f_{\alpha,\beta}(p) = p$ for $\alpha = 0$ and $\beta = 1$; these are the "neutral" values, $\mathbf{0} = (0, 1) \in \Theta$. Cox starts his exposition in [3, Section 3] from the one-parameter subfamily

$$f_\beta(p) := \frac{p^\beta}{p^\beta + (1-p)^\beta}, \tag{6}$$

where $\beta \in \mathbb{R}$, obtained by fixing $\alpha := 0$. We are mostly interested in $\beta > 0$, although $\beta = 0$ (when every p is transformed to 0.5) and $\beta < 0$ (which reverses the order of the label probabilities) are also possible. Similarly, we can set β to its neutral value 1 obtaining another one-parameter subfamily,

$$f_\alpha(p) := \frac{p \exp(\alpha)}{p \exp(\alpha) + (1-p)}. \tag{7}$$

An example of a function in the class (6) is shown in Figure 1 in green (with $\beta = 0.75$); the plot in orange shows a function in the class (7) (with $\alpha = 1$).

Remark 1. Cox's calibration functions look particularly natural (are linear functions) in terms of the log odds ratios (as presented in Cox [3, Section 3]). Namely, (5) can be rewritten as

$$\log \frac{f_{\alpha,\beta}(p)}{1 - f_{\alpha,\beta}(p)} := \alpha + \beta \log \frac{p}{1-p}. \tag{8}$$

Multiclass case

The advantage of the representations (5), (6), and (7) over (8) is that they are easy to extend to the multiclass case. Let the label space be $\mathbf{Y}$ with $K := |\mathbf{Y}| < \infty$; therefore, each prediction is a set of nonnegative numbers $\mathbf{p} = (p_y \mid y \in \mathbf{Y}) \in [0,1]^K$ summing to 1, where p_y is the predicted probability of y. The calibration functions (5) become

$$f_{\alpha,\beta}(\mathbf{p})_y := \frac{p_y^\beta \exp(\alpha(y))}{\sum_{y' \in \mathbf{Y}} p_{y'}^\beta \exp(\alpha(y'))}, \tag{9}$$

where $\alpha \in \mathbb{R}^{\mathbf{Y}}$ and $\beta \in \mathbb{R}$. Formally, the number of parameters in (9) is $K+1$, but the effective number of parameters is K since the transformation (9) does not change when the same constant (positive or negative) is added to all $\alpha(y)$, $y \in \mathbf{Y}$. The calibration functions (6) become

$$f_\beta(\mathbf{p})_y := \frac{p_y^\beta}{\sum_{y' \in \mathbf{Y}} p_{y'}^\beta},$$

where $\beta \in \mathbb{R}$. Therefore, this family still depends on one real-valued parameter, β.

3 Protecting prediction algorithms

For any predictive system, we define its *probability process* as a function mapping any finite sequence of observations to the product $B_{p_1}(y_1) \cdots B_{p_n}(y_n)$ (i.e., the probability attached to this sequence by the predictive system), where n is the number of observations in the sequence, $y_1, \dots, y_n$ are their labels, and $p_1, \dots, p_n$ are the predictions for those observations. We regard the probability process as the capital process of a player playing an extremely challenging (definitely unfair) game: his capital cannot go up, and for it not to go down he has to predict with the probability measure concentrated on the true outcome. The probability process of the base predictive system will be referred to as the *base probability process*.

Remark 2. The notion of a probability process is very similar to Cox's [4] notion of partial likelihood, but we cannot say that a probability process is partial in any sense (since it is not part of a fuller probability process: there is no probability measure on the objects [17, Section 10.5]). In the absence of objects it becomes close to likelihood (however, unlike likelihood, it is not a function of any parameters). It is even closer to the notion of measure as used in the algorithmic theory of randomness: see, e.g., [13, Definition 4.2.1].

For simplicity, we will discuss only positive probability processes and martingales (i.e., those that do not take zero values). This will be sufficient for the considerations of Section 5. Each test martingale with respect to the base predictive system is the ratio of a probability process to the base probability process

Algorithm 2 Composite Jumper predictor $((p_1, p_2, \dots) \mapsto (p'_1, p'_2, \dots))$

$P := \pi$
$A^J_\theta := \frac{1-\pi}{|\mathbf{J}|} 1_{\theta=0}$ for all $J \in \mathbf{J}$ and $\theta \in \Theta$
for $n = 1, 2, \dots$:
    for $J \in \mathbf{J}$:
        $A := \sum_\theta A^J_\theta$
        for $\theta \in \Theta$: $A^J_\theta := (1 - J)A^J_\theta + AJ/|\Theta|$
    $p'_n := p_n P + \sum_{J,\theta} f_\theta(p_n) A^J_\theta$
    $P := P B_{p_n}(\{y_n\})$
    for $J \in \mathbf{J}$ and $\theta \in \Theta$: $A^J_\theta := A^J_\theta B_{f_\theta(p_n)}(\{y_n\})$
    $C := P + \sum_{J,\theta} A^J_\theta$
    $P := P/C$
    for $J \in \mathbf{J}$ and $\theta \in \Theta$: $A^J_\theta := A^J_\theta / C$

(i.e., is a probability ratio process, familiar from sequential analysis), and vice versa. This establishes a bijection between test martingales and probability processes (for a fixed base predictive system). Algorithm 2 is the predictive system whose probability process corresponds to the test martingale of Algorithm 1 averaged as in (3).

Algorithm 2 implements the Bayesian merging rule and is a special case the Aggregating Algorithm (AA) [20] corresponding to the *log-loss function*

$$\lambda(y, p) := \begin{cases} -\log p & \text{if } y = 1 \\ -\log(1 - p) & \text{if } y = 0, \end{cases} \tag{10}$$

where $y \in \{0, 1\}$ is the true label and $p \in [0, 1]$ is its prediction (the logarithm is typically natural, but in Section 5 we will consider decimal logarithms). In the case where $\pi = 0$ and $|\mathbf{J}| = 1$, it is also a special case of the Fixed Share algorithm of [10].

The AA is described in [20, Section 2], and in our case of the log-loss function the optimal in a natural sense learning rate is $\eta := 1$, and the AA coincides with the APA ("Aggregating Pseudo-Algorithm"). Analogously to (2) but reflecting the operation of averaging (3), to each $J \in \mathbf{J}$ and each sequence $\boldsymbol{\theta} = (\theta_1, \theta_2, \dots)$ corresponds the *elementary predictor* that outputs, at each step n,

$$p'_n := f_{\theta_n}(p_n), \quad n = 1, 2, \dots,$$

as its prediction. There is also the *base* elementary predictor that just coincides with the base predictive system. The prior probability measure on the elementary predictors is built on top of the Simple Jumper Markov chain (described in the previous section) and taking the averaging (3) into account:

- With probability π, we choose the base elementary predictor (which is our *passive* elementary predictor).

- Otherwise, we choose the jumping rate J from the uniform probability measure on $\mathbf{J}$.
- Having chosen J, we generate $\boldsymbol{\theta} := (\theta_1, \theta_2, \dots)$ as in the previous section, which gives us the *active* elementary predictor $(J, \boldsymbol{\theta})$.

This determines the prior distribution on the elementary predictors.

The variable P in Algorithm 2 holds the posterior weight of the passive elementary predictor, and A_θ^J holds the total posterior weight of the elementary predictors $(J, \boldsymbol{\theta})$ that are in the state θ (i.e., $\theta_n = \theta$, where $\boldsymbol{\theta} = (\theta_1, \theta_2, \dots)$ and n is the current step). We initialize them to their prior weights (lines 1–2), and the recursion is given in lines 6, 8, 9, 11, and 12. Line 6 corresponds to the transition function of the Simple Jumper Markov chain with the jumping rate J, and lines 8–9 are the Bayesian weight updates. In line 10 we compute the total posterior weight of the elementary predictors, and in lines 11–12 we normalize the weights. In line 7 we compute the protected prediction as weighted average of the predictions produced by the elementary predictors.

We refer to the method exemplified by Algorithm 2 as *protected probabilistic classification*. We start from a base predictive system, design a way of gambling against it (a test martingale), and then "turn the tables" and use the test martingale as a protection against the kind of changes that the test martingale benefits from. If and when those changes happen, the product (*protected probability process*) of the base probability process and the test martingale outperforms the base probability process; equivalently, the *protected predictive system*, given by Algorithm 2 applied to a base predictive system, outperforms the base predictive system in terms of the log loss function.

We will use the notation

$$\mathrm{Loss}(p_1, \dots, p_n \mid y_1, \dots, y_n) := \sum_{i=1}^{n} \lambda(y_i, p_i)$$

for the cumulative log-loss of predictions $p_i \in [0,1]$ on labels $y_i \in \{0,1\}$, where λ is defined by (10). The protection provided by Algorithm 2, and similar procedures, has its price, since the loss suffered by the protected predictive system can be greater than the loss suffered by the base predictive system. The *price of protection* for Algorithm 2 is defined to be

$$\sup\bigl(\mathrm{Loss}(p'_1, \dots, p'_n \mid y_1, \dots, y_n) - \mathrm{Loss}(p_1, \dots, p_n \mid y_1, \dots, y_n)\bigr) = \sup\left(-\ln S_n\right), \quad (11)$$

where the sup is over all n, all object spaces $\mathbf{X}$, all base predictive systems (outputting predictions $p_1, p_2, \dots$), and all sequences of observations (with y_i as their labels). Of course, the definition given by the left-hand side of (11) is applicable to any system transforming predictions $p_1, p_2, \dots$ to predictions $p'_1, p'_2, \dots$; for Algorithm 2 we have an equivalent definition given by the right-hand side of (11), S being the Composite Jumper martingale. We are particularly interested in the case of a finite price of protection.

4 A theoretical guarantee

A simple performance guarantee for Algorithm 2 is given by the following result (proved in Appendix A).

Theorem 3. *The price of protection for Algorithm 2 is* $\log \frac{1}{\pi}$. *Besides, for any* n, *any jumping rate* J, *any sequence of observations, and any sequence* $f_{\theta_1}, \dots, f_{\theta_n}$ *of calibrating functions (from the family* $(f_\theta \mid \theta \in \Theta)$*),*

$$\begin{aligned}\operatorname{Loss}(p'_1, \dots, p'_n \mid y_1, \dots, y_n) \\ \le \operatorname{Loss}(f_{\theta_1}(p_1), \dots, f_{\theta_n}(p_n) \mid y_1, \dots, y_n) \\ + \log \frac{1}{1-\pi} + \log |\mathbf{J}| + k \log(|\Theta| - 1) \\ + k \log \frac{1}{J'} + (n - k - 1) \log \frac{1}{1 - J'}, \quad (12)\end{aligned}$$

where

$$J' := \frac{|\Theta| - 1}{|\Theta|} J \quad (13)$$

and $k = k(\theta_1, \dots, \theta_n)$ *is the number of switches,*

$$k := |\{i \in \{1, \dots, n\} : \theta_i \neq \theta_{i-1}\}|,$$

with $\theta_0 := \mathbf{0}$.

The price of protection $\log \frac{1}{\pi}$ in Theorem 12 is small unless π is very close to 0, even for test sets of a moderate size. For example, setting $\pi := 0.5$ appears a reasonable compromise between the two terms involving π in the price of protection and in (12), and we will use it in our experiments in the next section.

The value J' introduced in (13) is an alternative parameterization of the jumping rate (and it is used in, e.g., [20] as the main one); it is usually close to (or at least has the same order of magnitude as) J. We may call J' the effective jumping rate: it is the probability that the state actually changes at a given step.

The *regret term* in the last two lines of (12) can be interpreted as follows. First, it gives the degree to which we are competitive with an "oracular" predictive system that knows in advance which calibration function should be used at each step. We have already discussed the addend $\log \frac{1}{1-\pi}$, and $\log |\mathbf{J}|$ is the price, typically very moderate ($\log 3$ in our experiments), that we pay for using several jumping rates. The following two addends in the regret term give us the price, in terms of the log loss, for each switch. Namely, each switch costs us $\log(|\Theta| - 1)$ (which is $\log 8$ or $\log 20$ in our experiments) plus an amount that depends on the switching rate, namely $\log \frac{1}{J'}$. The last term in (12) is close, assuming $k \ll n$ and $J' \ll 1$, to nJ'. A reasonable choice of J' is the inverse $1/N$ of the expected number N of observations in the test set (but remember that Algorithm 2 covers a range of J, which is motivated by N typically not being known in advance). With this choice the price $\log \frac{1}{J'}$ to pay per switch becomes $\log N$.

Remark 4. The Markov chains usually used in tracking the best expert (see, e.g., [20, Section 3.3]) choose the first state randomly with equal probabilities, whereas our Simple Jumper Markov chain starts from the neutral state $\mathbf{0} \in \Theta$. This expresses the "presumption of innocence" towards the base predictive system: we do not calibrate it unless calibration is needed (and we definitely do not calibrate it at the very beginning of the test set). The two approaches lead to extremely similar results in our experiments.

5 Experimental results

There are not many datasets suitable for our experiments. First, they should be ordered chronologically (to fit the scenario of Section 1), or at least contain timestamps for all observations. And second, they should not be of the time-series type, so that applying typical machine-learning prediction algorithms (such as those implemented in `scikit-learn`) makes sense.

Our main dataset will be `Bank Marketing` (the only dataset in the top twelve most popular datasets at the UC Irvine Machine Learning Repository that satisfies our requirements; we will use, however, the full version of this dataset as given at the `openml.org` repository, which is easier in `scikit-learn`). The dataset consists of 45,211 observations representing telemarketing calls for selling long-term deposits offered by a Portuguese retail bank, with data collected from 2008 to 2013 [14]. The labels are 1 or 2, which we encode as 0 and 1, respectively (and we will never mention the original labels again in this paper). Label 1 indicates a successful sale, and such observations comprise only 12% of all labels.

The observations are listed in chronological order. We take the first 10,000 observations as the training set, normalize the attributes of the objects using `StandardScaler` in `scikit-learn` (although normalization barely affects our results), and train Random Forest with default parameters and random seed 2021 on it. (In our figures in this section we use Random Forest as the base prediction algorithm, since it consistently produces good results in our experiments.) Random Forest often outputs probabilities of success that are equal to 0 or 1, and when such a prediction turns out to be wrong (which happens repeatedly), the log-loss is infinite. It is natural, therefore, to truncate a probability $p \in [0,1]$ of 1 to the interval $[\epsilon, 1-\epsilon]$ replacing p by

$$p^* := \begin{cases} \epsilon & \text{if } p \le \epsilon \\ p & \text{if } p \in (\epsilon, 1-\epsilon) \\ 1-\epsilon & \text{if } p \ge 1-\epsilon, \end{cases} \tag{14}$$

where we set $\epsilon := 0.01$ (in `scikit-learn`, $\epsilon = 10^{-15}$, but it appears excessive to us). The resulting prediction model is our base predictive system. After we find it, we never use the training set again, and the numbering of observations starts from the first element of the test set (i.e., the dataset in the chronological order without the training set).

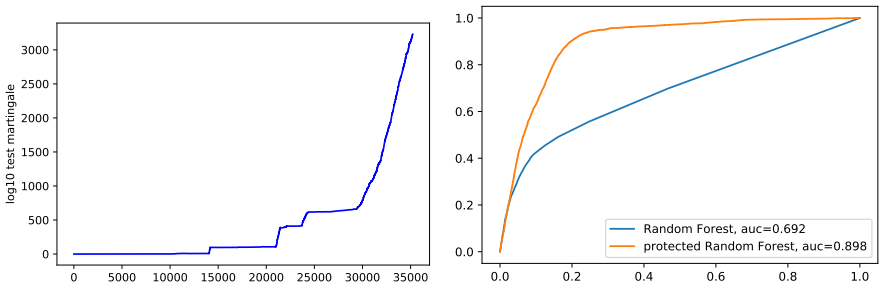

Figure 2: Left panel: The Composite Jumper test martingale for the `Bank Marketing` dataset and Random Forest. Right panel: The ROC curve for the Composite Jumper protection.

Table 1: The AUC for the `Bank Marketing` dataset and key `scikit-learn` functions (with default parameters) together with their protected versions.

Prediction algorithm	**base**	**protected**
Random Forest	0.692	0.898
Gradient Boosting	0.734	0.901
Decision Trees	0.564	0.814
Neural Network	0.665	0.879
SVM	0.686	0.844
Naive Bayes	0.646	0.807
Logistic Regression	0.609	0.838

The left panel of Figure 2 shows the trajectory of $\log_{10} S_n$, $n = 1, \dots, 35211$, where S_n is the value of the Composite Jumper test martingale over the test set with the jumping rates $\mathbf{J} := \{10^{-2}, 10^{-3}, 10^{-4}\}$ and the family (5) with $\alpha \in \{-1, 0, 1\}$ and $\beta \in \{0.5, 1, 2\}$. With this choice of the ranges of α and β, which we always use in the binary case, there are $|\Theta| = 9$ of parameter vectors $\theta := (\alpha, \beta)$. The final value of the test martingale in the left panel of Figure 2 is approximately $10^{3231.7}$.

The right panel of Figure 2 gives the ROC curve for Random Forest and Random Forest protected by Algorithm 2. We can see that the improvement is substantial. In terms of the log-loss function and decimal logarithms, the loss goes down from 7185.1 to 3953.4 (the difference between these two numbers being, predictably, the exponent 3231.7 in the final value of the test martingale in the left panel).

Table 1 gives the AUC (area under curve) for Algorithm 2 and several base prediction algorithms implemented in `scikit-learn`. Since the dataset is imbalanced, AUC is a more suitable measure of quality than error rate.

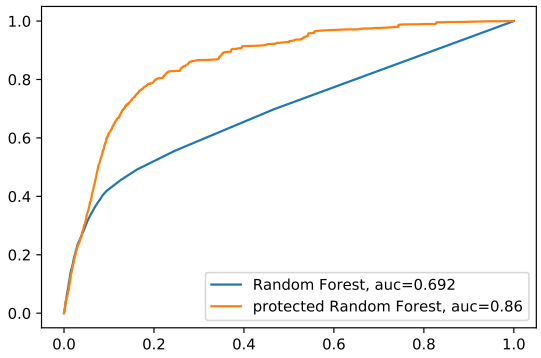

Figure 3: The ROC curve for the `Bank Marketing` dataset and Random Forest with the Composite Jumper protection and feedback provided for every 100th test observation.

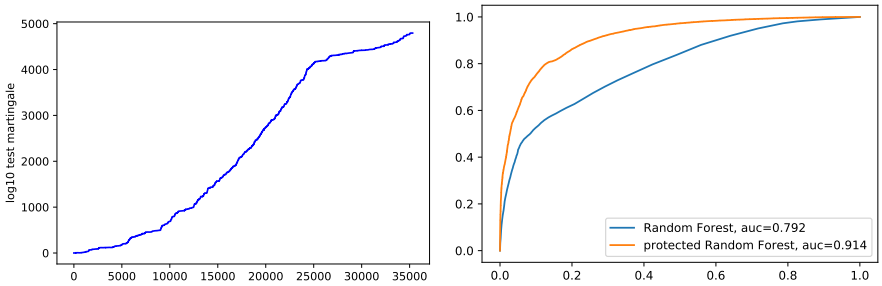

Figure 4: The analogue of Figure 2 for the `electricity` dataset.

In our experiments so far we have assumed that every test observation is used for recalibrating the prediction model. From the practical point of view this may be unrealistic, and in reality we can only hope to get feedback on a fraction of the test observations. Figure 3 is the counterpart of the right panel of Figure 2 for the case where the vast majority of observations are predicted by the prediction model without getting any feedback, and the weight updates in Algorithm 2 are run only on every 100th test observation (so that the same weights P and A_θ^J are used for the observations $100k+1$, $100k+2$,..., $100k+100$ for $k = 0, 1, \dots$). Now the improvement is more modest.

Another similar dataset is `electricity` [7, 9], available from `openml.org`. Its 45,312 observations contain binary labels (whether the electricity price in New South Wales goes up or down, encoded as 1 and 0, respectively) together with relevant attributes, collected from 7 May 1996 to 5 December 1998. The observations are listed in the chronological order, as for `Bank Marketing`, and we use the same scheme, allocating the first 10,000 observations to the train-

Table 2: Numbers of errors for the `electricity` dataset and some `scikit-learn` functions (with default parameters) together with their protected versions.

Prediction algorithm	**base**	**protected**
Random Forest	9184	5846
Gradient Boosting	9165	6009
Decision Trees	9372	6806
Neural Network	14,358	7469
SVM	14,433	9532

ing set and normalizing the attributes with `StandardScaler`; the figures use the same base prediction algorithm (Random Forest). Figure 4 shows results for this dataset; protection still greatly improves the performance of the base predictions.

Table 2 gives results for Algorithm 2 and several base prediction algorithms (`scikit-learn`'s Naive Bayes and Logistic Regression fail on this dataset producing inconsistent results). Now the dataset is balanced, and so we report the numbers of errors (i.e., the cases of the predictions being different from the true labels).

Multiclass case

Here we work with the `UJIIndoorLoc` dataset [19], available from the UC Irvine Machine Learning Repository. The attributes are intensity levels of 520 wireless access points (WAPs) in three buildings of the Jaume I University in Castelló de la Plana, Valencia, Spain. The task we are interested in is to identify the building given the WAP intensity levels. The dataset consists of two parts, a sizable original training set and a much smaller original validation set (the latter collected 4 months after the former). Since the attributes that we use are given on the same scale, there is no need to normalize them.

We consider two scenarios:

- In Scenario 1, we ignore the original validation set and use only the original training set, which we order chronologically and then split into the training set consisting of the first 10,000 observations and the test set consisting of the remaining observations, as we did for `Bank Marketing` and `electricity`.
- In Scenario 2, we use the original training set as our training set and the original validation set as our test set.

In this multiclass case we truncate a probability measure $p = (p_y)$ slightly

Table 3: Numbers of errors in Scenario 1 for the `UJIIndoorLoc` dataset and key `scikit-learn` functions (with default parameters).

Prediction algorithm	**base**	**protected**
Random Forest	556	208
Gradient Boosting	1397	1041
Decision Trees	1185	1185
Neural Network	1289	1261
SVM	354	199
Naive Bayes	43	43
Logistic Regression	290	244

differently from the binary procedure (14); namely, we set

$$p^*_y := \frac{\max(p_y, \epsilon)}{\sum_{y' \in \mathbf{Y}} \max(p_{y'}, \epsilon)}, \quad y \in \mathbf{Y} \tag{15}$$

(where $\mathbf{Y}$ are the labels standing for the three buildings, $|\mathbf{Y}| = 3$), and we still set $\epsilon := 0.01$.

First we report our results for the more difficult Scenario 1. It is interesting that one of the buildings is not in the test set; it seems that the buildings were explored systematically. The results for Scenario 1 are given in Table 3, where β range over $\{0.5, 1, 2\}$ and α range over the 7 binary vectors of length 3 apart from $(1, 1, 1)$ (which is not included since the corresponding calibrating function is the same as for $(0, 0, 0)$); of course, the neutral calibrating function is the one with $\alpha = (0, 0, 0)$ and $\beta = 1$.

Protection always improves results, sometimes significantly, apart from Decision Trees and Naive Bayes, for which the number of errors stays the same, 1185 and 43, respectively. In the case of Decision Trees, the vast majority of the base predictions are categorical, concentrating on one label, which makes their calibration problematic. (Namely, 9251 out of 9937 predictions are categorical, assigning probability 1 to one of the labels; in particular, all wrong predictions are categorical. The least categorical of the remaining predictions assigns probability 0.974 to one of the labels.) In the case of Naive Bayes, the quality of the probabilistic predictions still improves greatly: the log10 loss of the base predictive system is ∞ because the loss is ∞ for 8 observations; if those 8 observations are ignored, the log10 loss is 2304.76, whereas the log10 loss of the protected predictive system is 84.41 (without any infinities).

Scenario 2 is extremely easy (since all three buildings have been explored completely); e.g., Logistic Regression as implemented in `scikit-learn` does not make any errors. For the results, see Table 4. It is reassuring that the number of errors never goes up as result of protection.

Table 4: Numbers of errors in Scenario 2 for the `UJIIndoorLoc` dataset and `scikit-learn` functions with default parameters.

Prediction algorithm	**base**	**protected**
Random Forest	2	1
Gradient Boosting	2	2
Decision Trees	21	10
Neural Network	1	1
SVM	4	4
Naive Bayes	9	9
Logistic Regression	0	0

6 Conclusion

The methods of adaptive calibration that we propose in this paper need to be validated on other datasets and perhaps for other calibrating functions. Notice that calibrating functions may depend not only on the current predicted probability p but also on the current object x. (So that "calibration" may be understood in a very wide sense, as in [5], and include elements of "resolution" [6].)

In this paper we only use the log loss function. Arguably it is the most fundamental one [22], but its disadvantage is that, for many base prediction algorithms, we need truncation (see (14) and (15)) to prevent an infinite loss. A popular alternative to the log loss function is the Brier loss function [2]; it is much more forgiving and does not require truncation. It follows from the results of [25] that Theorem 3 will continue to hold when the log loss function is replaced by the Brier loss function. Empirical studies of the performance of our procedures in this case are an interesting direction of further research.

A useful feature of our procedure of protection is that it is cheap, which is achieved by mixing Simple Jumper martingales with constant 1: see (3). The role of mixing with 1 is to insure against a catastrophic loss of evidence against the null hypothesis (given by the base predictive system) found by those test martingales. There are much more sophisticated ways of insuring against loss of evidence [17, Chapter 11], and they will provide further protection.

Acknowledgments

We are grateful to Glenn Shafer for his advice. Phelype Oleinik has helped us with TEX. This research has been partially supported by Amazon and Stena Line. In our computational experiments we have used `scikit-learn` [16].

References

[1] Francis J. Anscombe. Rejection of outliers. *Technometrics*, 2:123–147, 1960.

[2] Glenn W. Brier. Verification of forecasts expressed in terms of probability. *Monthly Weather Review*, 78:1–3, 1950.

[3] David R. Cox. Two further applications of a model for binary regression. *Biometrika*, 45:562–565, 1958.

[4] David R. Cox. Partial likelihood. *Biometrika*, 62:269–276, 1975.

[5] A. Philip Dawid. Calibration-based empirical probability (with discussion). *Annals of Statistics*, 13:1251–1285, 1985.

[6] A. Philip Dawid. Probability forecasting. In Samuel Kotz, N. Balakrishnan, Campbell B. Read, Brani Vidakovic, and Norman L. Johnson, editors, *Encyclopedia of Statistical Sciences*, volume 10, pages 6445–6452. Wiley, Hoboken, NJ, second edition, 2006.

[7] João Gama, Pedro Medas, Gladys Castillo, and Pedro Rodrigues. Learning with drift detection. In Ana L. C. Bazzan and Sofiane Labidi, editors, *Advances in Artificial Intelligence: SBIA 2004*, pages 286–295, Berlin, 2004. Springer.

[8] I. J. Good. Rational decisions. *Journal of the Royal Statistical Society B*, 14:107–114, 1952.

[9] Michael Harries. Splice-2 comparative evaluation: Electricity pricing. Technical Report UNSW-CSE-TR-9905, Artificial Intelligence Group, School of Computer Science and Engineering, University of New South Wales, July 1999.

[10] Mark Herbster and Manfred K. Warmuth. Tracking the best expert. *Machine Learning*, 32:151–178, 1998.

[11] Peter J. Huber and Elvezio M. Ronchetti. *Robust Statistics*. Wiley, Hoboken, NJ, second edition, 2009.

[12] Daniel Kahneman, Paul Slovic, and Amos Tversky, editors. *Judgment under Uncertainty: Heuristics and Biases*. Cambridge University Press, Cambridge, 1982.

[13] Ming Li and Paul Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer, fourth edition, 2019.

[14] Sérgio Moro, Paulo Cortez, and Paulo Rita. A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, 62:22–31, 2014.

[15] James R. Norris. *Markov Chains*. Cambridge University Press, Cambridge, 1997.

[16] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[17] Glenn Shafer and Vladimir Vovk. *Game-Theoretic Foundations for Probability and Finance*. Wiley, Hoboken, NJ, 2019.

[18] Albert N. Shiryaev. *Probability-2*. Springer, New York, third edition, 2019.

[19] Joaquín Torres-Sospedra, Raúl Montoliu, Adolfo Martínez-Usó, Joan P. Avariento, Tomás J. Arnau, Mauri Benedito-Bordonau, and Joaquín Huerta. `UJIIndoorLoc`: A new multi-building and multi-floor database for WLAN fingerprint-based indoor localization problems. In *2014 International Conference on Indoor Positioning and Indoor Navigation (IPIN 2014)*, pages 261–270. Institute of Electrical and Electronics Engineers, 2014.

[20] Vladimir Vovk. Derandomizing stochastic prediction strategies. *Machine Learning*, 35:247–282, 1999.

[21] Vladimir Vovk. Competitive on-line statistics. *International Statistical Review*, 69:213–248, 2001.

[22] Vladimir Vovk. The fundamental nature of the log loss function. In Lev D. Beklemishev, Andreas Blass, Nachum Dershowitz, Berndt Finkbeiner, and Wolfram Schulte, editors, *Fields of Logic and Computation II: Essays Dedicated to Yuri Gurevich on the Occasion of His 75th Birthday*, volume 9300 of *Lecture Notes in Computer Science*, pages 307–318, Cham, 2015. Springer.

[23] Vladimir Vovk. Testing randomness online, On-line Compression Modelling project (New Series), `http://alrw.net`, Working Paper 24, June 2019. Journal version: *Statistical Science* 36:595–611, 2021.

[24] Vladimir Vovk. Protected probabilistic regression, On-line Compression Modelling project (New Series), `http://alrw.net`, Working Paper 34, May 2021.

[25] Vladimir Vovk and Fedor Zhdanov. Prediction with expert advice for the Brier game. *Journal of Machine Learning Research*, 10:2445–2471, 2009.

A Some proofs

The second statement of Theorem 3 can be deduced from the fact that the probability process of the Bayes mixture is equal to the average of the elementary predictors' probability processes with respect to the prior probability measure on the elementary predictors. This can be checked directly and is a special case of a known result for the AA specialized to the log loss function. In the context of this special case of the AA, a probability process is a process of the form $\exp(-L)$, where L is a loss process. The following lemma is the main property of the AA in this context.

Lemma 5. *The probability process of the AA with the log loss function is the average of the elementary predictors' probability processes with respect to the prior probability measure.*

Proof. For a proof, see [21, Lemma 1]. □

Proof of Theorem 3

Let us first prove that the price of protection is $\log \frac{1}{\pi}$. It is obvious that it does not exceed $\log \frac{1}{\pi}$, and it suffices to prove that, for any $J \in \mathbf{J}$, the likelihood ratio of the Simple Jumper predictor with jumping rate J to the base predictive system tends to 0 for some sequence of observations. We will prove more: namely, for a suitable choice of the base predictive system, the likelihood ratio of the Simple Jumper predictor with jumping rate J to the base predictive system tends to 0 a.s. under the probability distribution of the base predictive system.

By the ergodic theorem for Markov chains (see, e.g., [15, Theorem 1.10.2]) the Simple Jumper Markov chain will spend, asymptotically and almost surely, the fraction $1/|\Theta|$ of its time in each state $\theta \in \Theta$. Choose a p such that $f_\theta(p) \neq p$ for some $\theta \in \Theta$. Let the base predictive system always output p. By Kabanov et al.'s criterion (see, e.g., [18, Theorem 4]) of mutual singularity of probability measures in "predictable terms", the Simple Jumper predictive system with jumping rate J and the base predictive system will be mutually singular. This implies (by [18, Theorem 2]) that, indeed, the likelihood ratio of the Simple Jumper to the base predictive system tends to 0 a.s.

To complete the proof of Theorem 3, notice that the probability that the Simple Jumper Markov chain with a given jumping rate $J \in \mathbf{J}$ produces $\theta_1, \dots, \theta_n$ with k switches as its first n states is

$$\left(\frac{J'}{|\Theta| - 1}\right)^k (1 - J')^{n-k-1}.$$

Therefore, the protected probability process is at least

$$\frac{1 - \pi}{|\mathbf{J}|} \left(\frac{J'}{|\Theta| - 1}\right)^k (1 - J')^{n-k-1}$$

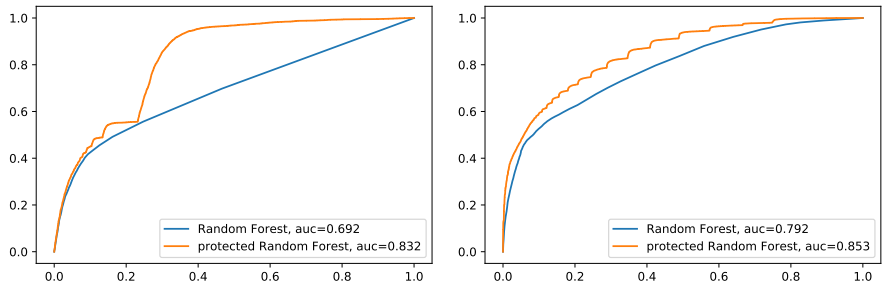

Figure 5: The ROC curves for the `Bank Marketing` (left panel) and `electricity` (right panel) dataset and Random Forest with the Composite Jumper protection based on quadratic calibration.

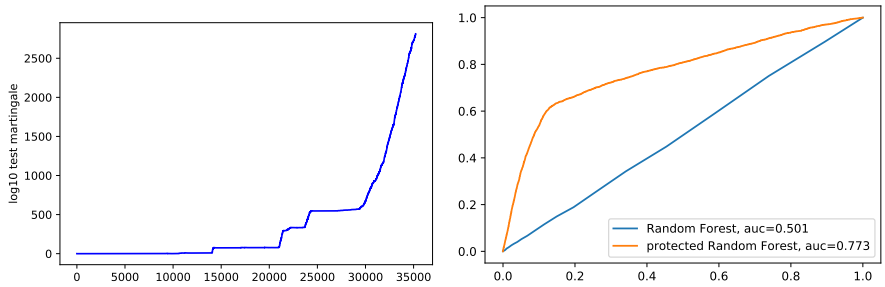

Figure 6: The analogue of Figure 2 for the `Bank Marketing` dataset with randomly permuted objects.

times the probability process for any elementary predictor with jumping rate J and the sequence of states beginning with $\theta_1, \dots, \theta_n$. It remains to convert the probability processes into log loss processes by applying the operation $-\log$.

B Further experimental results

We start this appendix from reporting results for the family (1) of quadratic calibrating functions with θ restricted to a finite set Θ. As always, we choose a minimal Θ, namely $\Theta := \{-1, 0, 1\}$. Figure 5 is the counterpart of the right panels of Figures 2 and 4 for this family. In the rest of the paper we only consider Cox's calibrating functions.

Next we discuss some of the reasons for a good performance of our protection procedures on the `Bank Marketing` and `electricity` datasets. It is revealing that protection still gives good results for both datasets when the objects are randomly shuffled (so that they become uninformative). See Figures 6 and 7.

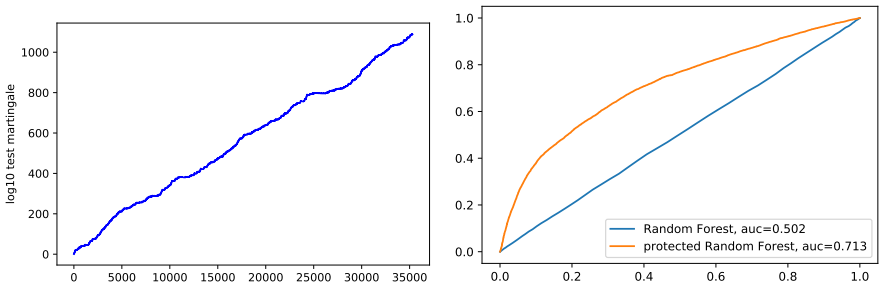

Figure 7: The analogue of Figure 4 for the `electricity` dataset with randomly permuted objects.

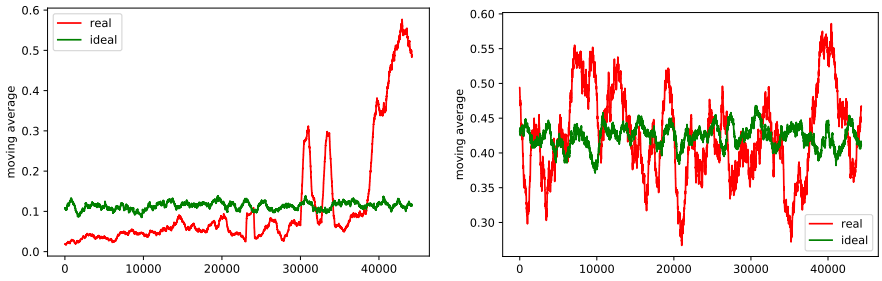

Figure 8: The moving averages of the labels for the `Bank Marketing` (left panel) and `electricity` (right panel) datasets, as described in text.

Therefore, already the order of the test labels is informative. Of course, the ROC curves for the unprotected procedures become trivial (close to the main diagonal) when the objects are shuffled.

To understand the reasons for the ROC curves being non-trivial after protection in Figures 6 and 7, we compute the moving averages of the labels for the two datasets (including both training and test sets). Figure 8 shows in red the trajectories of the moving averages of the labels: the value of each trajectory at time n is the arithmetic mean of the 1000 consecutive labels starting from y_n (namely, the arithmetic mean of $y_n, \dots, y_{n+999}$). For comparison, the analogous moving average for a simulated IID binary sequence with the right percentage of 1s (12% for `Bank Marketing` and 42% for `electricity`) is shown in green.

The behaviour of the moving average is particularly anomalous for `Bank Marketing`: the proportion of successful calls increases drastically towards the end of the dataset, which explains the quick growth of the Composite Jumper martingale in Figures 2 and 6 starting from approximately the 30,000th observation in the test set, which corresponds to the 40,000th observation in the full dataset. The percentage of successful calls is 3.5% for the training set and 14.0%

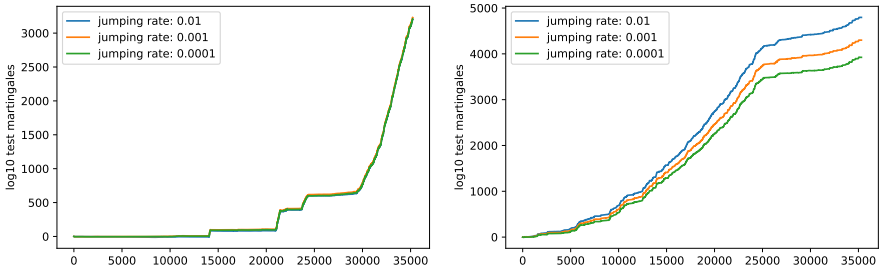

Figure 9: The Simple Jumper martingales for various jumping rates for the `Bank Marketing` (left panel) and `electricity` (right panel) datasets, as described in text.

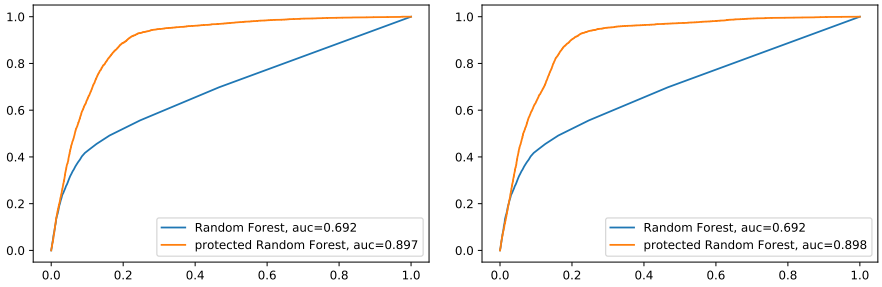

Figure 10: The ROC curves for the `Bank Marketing` dataset and Random Forest with the Simple Jumper protection for the jumping rate $J := 10^{-2}$ on the left and $J := 10^{-4}$ on the right.

for the test set. The behaviour for `electricity` is less anomalous, but there are still clear non-random patterns.

Dependence on the jumping rate

The left panel of Figure 9 shows the dependence of the Simple Jumper martingale (with the same parameters as in the left panel of Figure 2) on the jumping rate for the `Bank Marketing` dataset; the dependence is slight, at least on the log scale. The right panel is its counterpart for the `electricity` dataset; the dependence on the jumping rate is more noticeable.

The dependence of the resulting ROC curves on the jumping rate is also weak: see, e.g., Figure 10, which shows results for the jumping rates 0.01 and 0.0001. However, using a specific value of the jumping rate J may be risky in that the Simple Jumper test martingale loses capital exponentially quickly if the base prediction algorithm is already ideal; this will make the insurance policy discussed in Section 1 expensive both for testing and for prediction. A

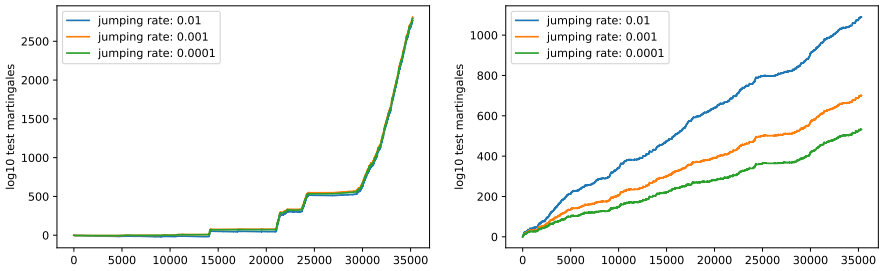

Figure 11: The Simple Jumper martingales for the `Bank Marketing` and `electricity` dataset with randomly permuted objects.

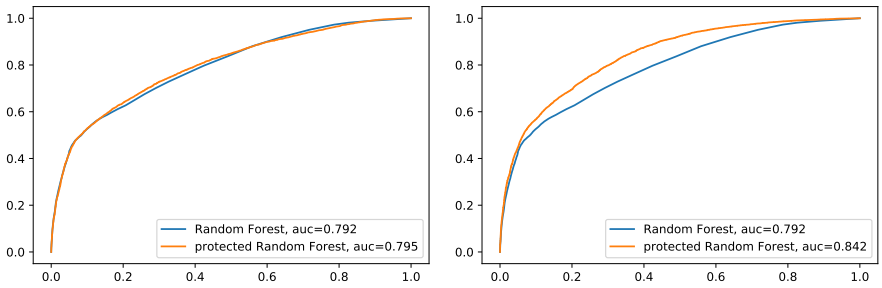

Figure 12: Left panel: The ROC curve for the `electricity` dataset and Random Forest with the Composite Jumper protection and feedback provided for every 100th test observation. Right panel: the counterpart of the left panel for feedback provided for every 10th observations.

safer option is to use the Composite Jumper procedures, as we do in the main paper.

Figure 11 showing the Simple Jumper martingales for various jumping rates is interesting because of the striking difference between the `Bank Marketing` and `electricity` datasets with randomly permuted objects.

Limited feedback

The left panel of Figure 12 is the counterpart for the `electricity` dataset of Figure 3: feedback is provided only for every 100th test observation. Now protection with limited feedback results only in marginal improvement in the AUC for the ROC curve. With the fuller feedback comprising every 10th test observation the improvement is again substantial: see the right panel of Figure 12.

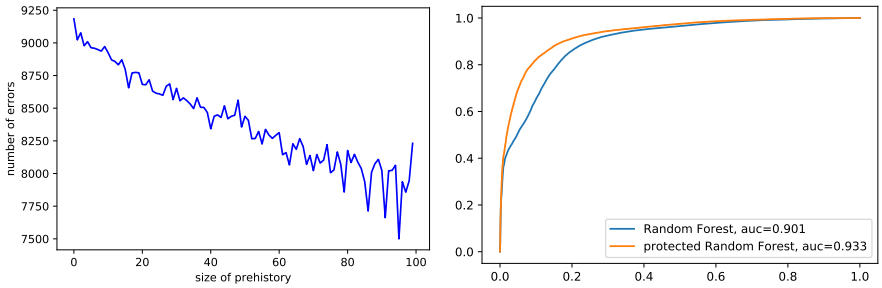

Figure 13: Left panel: The number of errors made by Random Forest on the `electricity` dataset with added prehistory. Right panel: The ROC curve for the `electricity` dataset with prehistory of 48 and Random Forest with the Composite Jumper protection.

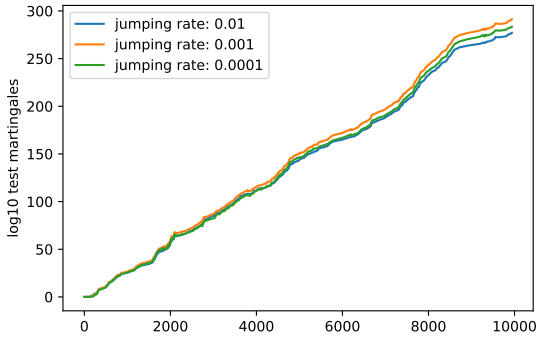

Figure 14: The Simple Jumper martingales for the `UJIIndoorLoc` dataset (Scenario 1) and Random Forest.

The time-series aspects of the `electricity` dataset

Unlike the two other datasets considered in this paper, the `electricity` dataset consists of periodic observations referring to a period of 30 minutes, so that there are 48 instances for each time period of one day. Therefore, complementing the existing attributes of each observation by a prehistory, i.e., the labels of a given number (the *size of prehistory*) of immediately preceding observations, may improve the predictions. The left panel of Figure 13 shows that this is indeed the case. The right panel of Figure 13 shows the improvement in the ROC curve for the prehistory of size 48 (one day).

Multiclass case

Figure 14 gives the trajectories of the three components of the Composite Jumper martingale with $\mathbf{J} = \{10^{-2}, 10^{-3}, 10^{-4}\}$ and $\Theta = (\{0,1\}^3 \setminus \{(1,1,1)\}) \times \{0.5, 1, 2\}$ (as before) based on Random Forest for the `UJIIndoorLoc` dataset in Scenario 1. On the log scale, the three components do not appear vastly different, but the final value for the jumping rate 0.001 is more than 10^6 times larger than the final value for the jumping rate 0.01. The trajectory (not shown) for the Composite Jumper martingale with these jumping rates and $\pi = 0.5$ is visually indistinguishable from the highest of the three trajectories that are shown (the one for the jumping rate 0.001).